JMLR: Workshop and Conference Proceedings 25:175–190, 2012 Asian Conference on Machine Learning

More Is Better: Large Scale Partially-supervised Sentiment Classification

Yoav Haimovitch YOAVH@CS.TECHNION.AC.IL
Department of Computer Science, The Technion, Haifa, 32000 Israel

Koby Crammer KOBY@EE.TECHNION.AC.IL
Shie Mannor SHIE@EE.TECHNION.AC.IL
Department of Electrical Engineering, The Technion, Haifa, 32000 Israel

Editor: Steven C.H. Hoi and Wray Buntine

Abstract

We describe a bootstrapping algorithm to learn from partially labeled data, and the results of an empirical study for using it to improve performance of sentiment classification using up to 15 million unlabeled Amazon product reviews. Our experiments cover semi-supervised learning, domain adaptation and weakly supervised learning. In some cases our methods were able to reduce test error by more than half using such large amount of data.

Keywords: Sentiment Analysis, Semi-Supervised Learning, Domain Adaptation, Weakly Supervised Learning

1. Introduction

Sentiment analysis (Nasukawa and Yi, 2003; Chen et al., 2011; Bai, 2011; Prabowo and Thelwall, 2009; Pang and Lee, 2005) is the task of extracting opinions and emotions in general, and from a given text, such as articles or product reviews, in particular. Performing this analysis automatically can be used in many applications, such as generating reports for large companies about their products from the practically limitless amount of data that is available online.

A fundamental task of sentiment analysis is sentiment classification—given a review about a product the goal is to classify whether it is positive or negative with respect to the subject of the review. Pang and Lee (2008) mention that a majority of end-users claim that they are influenced by online reviews, and in fact actively search for them. Aggregating the ever growing amounts of sentimental data may benefit consumers as well.

In our research we follow Blitzer et al. (2007) and use Amazon product reviews, together with the associated rating given by the reviewer. Unlike most, if not all, previous work, we scale-up our study, and use up to $15M$ unlabeled reviews.

We describe a bootstrapping (self training) algorithm and apply it in a semi-supervised learning setting, where only very small amount of labeled data is available. We show that our method can reduce the test error by about 40% relatively to a model built with only $1,000$ labeled examples, and further investigate the influence of the initial size of the training set, and various algorithmic choices, to obtain an optimal algorithmic combination. Motivated from these results we apply our algorithm to two more tasks: domain adaptation with

more than 30 domains and, unsupervised learning that replaces labeled documents with simple rules designed with prior knowledge. We show that in all problems our algorithm is improving performance using this large amount of unlabeled data.

2. Supervised Binary Classification

Our algorithm reduces learning with partially labeled data to supervised learning, which we describe now. Each review is represented with a vector $\boldsymbol{x} \in \{0,1\}^F$ where for our data $F \approx 33M$ is the number of features, and is associated with a binary label $y \in \{\pm 1\}$, where $y = 1$ ($y = -1$) is associated with positive (negative) sentiment. We focus on linear models: functions of the form $\text{sign}(\boldsymbol{w} \cdot \boldsymbol{x})$ for some weight vector $\boldsymbol{w} \in \mathbb{R}^F$.

We assume the existence of a supervised algorithm that given a labeled set $S_l = \{(\boldsymbol{x}_i, y_i)\}$ outputs a model $\boldsymbol{w}$ that performs well on that set. That is, the fraction of reviews from S_l for which $\boldsymbol{w}$ outputs a wrong label is small. We chose to use the AROW algorithm (Adaptive Regularization of Weight vectors) of Crammer et al. (2009) since it was shown to work well on binary document classification, in general, and sentiment classification in particular. The AROW algorithm is based on the CW (confidence weighted) framework (Dredze and Crammer, 2008).

The algorithm is incremental, or online, and works in rounds. Often, it is initialized with the zero weight vector $\boldsymbol{w} = \boldsymbol{0} \in \mathbb{R}^F$. On each round AROW picks an example from the labeled set $(\boldsymbol{x}, y) \in S_l$, and uses it to update its current model $\boldsymbol{w}$. The algorithm maintains not only a weight vector $\boldsymbol{w} \in \mathbb{R}^F$ but also a diagonal matrix $\Sigma \in \mathbb{R}^{F \times F}$ both represent the mean and covariance of a Gaussian distribution. The algorithm updates both weight-vector and covariance after processing any example. Two parameters controls the behavior of the algorithm when it is executed in a batch setting: a learning rate r and the number of rounds the algorithm goes over the training set. We denote by $\boldsymbol{w} = AROW(S)$ the model that AROW outputs after iterating the training set S exactly once, setting the learning rate to $r = 10^{-5}$, which was fixed in all our experiments below. It is worth noting that a version where r is chosen dynamically quickly converages on $r = 10^{-5}$ across many settings, with very similar results to a fixed value (a fixed value saves running time).

3. Semi-Supervised Learning (SSL)

In semi-supervised learning, algorithms are introduced not only to a labeled set of example S_l, but also to an additional unlabeled set S_u. The new set contains only input vectors (or feature vectors) with no labels. The goal of the learning algorithm is to build a classifier $\boldsymbol{w}$ based on both resources, which is often called inductive semi-supervised learning. We denote the size of a set S by $|S|$. Typically $|S_l| \ll |S_u|$, as it is often easy or cheap to obtain unlabeled documents, yet labeling them is a long and expensive process. The goal of the learning algorithm is to improve the performance of the output model by incorporating S_u as well.

4. Algorithm

We propose to use a bootstrapping approach for SSL. Our algorithm first builds a model $\boldsymbol{w}^0$ using the labeled data $S_l^0 = S_l$, then using this model it picks a small subset A_N^0 of size

N from the unlabeled set S_u^0 and labels its inputs using the model $\boldsymbol{w}^0$, constructing a new labeled set,

$$S_l^1 \leftarrow S_l^0 \cup \left\{(\boldsymbol{x}, y) \ : \ \boldsymbol{x} \in A_N^0 \ , \ y = \text{sign}(\boldsymbol{w}^0 \cdot \boldsymbol{x})\right\} \ .$$

The algorithm then removes the newly labeled vectors from the unlabeled set, reducing its size by N, $S_u^1 \leftarrow S_u^0 / A_N^0$.

The algorithm works in iterations. On the i*th* iteration, it uses AROW to build a model $\boldsymbol{w}^i$ based on the set S_l^i of size $|S_l| + i \times N$, which is then used to choose and label a new set A_N^i of size N from the unlabeled set S_u^i of size $S_u - i \times N$. This set A_N^i is then labeled using $\boldsymbol{w}^i$ and removed from the unlabeled set S_u^{i+1} to the labeled set S_l^{i+1}. The algorithm stops when the labeled set is exhausted, i.e., a round i for which $S_u^i = \emptyset$[1]. For completeness, we include below plots of the error rate vs. the amount of unlabeled data that was labeled, evaluating the error rate in case of any-time early stopping.

To fully describe our algorithm, it remains to define how to choose the subset A_i^N. The algorithm uses the current model $\boldsymbol{w}^i$ to assign a score $s(\boldsymbol{x})$ to each unlabeled example $\boldsymbol{x} \in S_u^i$ and picks the N inputs with the highest score value. Since our learning algorithm employs linear models, a natural quantity to use is the distance of an input point $\boldsymbol{x}$ from the hyperplane defined by $\boldsymbol{w}$,

$$s(\boldsymbol{x}) = |\boldsymbol{x} \cdot \boldsymbol{w}| / \|\boldsymbol{w}\| \propto |\boldsymbol{w} \cdot \boldsymbol{x}| \ .$$

This approach was used by Tong and Koller (2001) in the context of active learning, where an input with the *lowest* score $s(\boldsymbol{x})$ is chosen to be labeled.

Finally, since AROW is an online algorithm, its output depends on the order of the inputs examples. We found that the best tradeoff between speed, simplicity and diversity is to fix a random permutation over S_l^0 and to add the new set of examples A_N^i, labeled by the current model, *before* S_l^i. In other words, in the next round $i+1$, AROW first learns with the recently added examples A_N^i and only then it learns with remaining labeled examples. The algorithm is summarized in Alg. 1.

We conclude this section with a complexity analysis of the algorithm. Let D be the maximum number of non-zero elements of inputs, $D = \max_{\boldsymbol{x}} \|\boldsymbol{x}\|_0$. Denote by $L = |S_l|$ and $U = |S_u|$ the size of the labeled set and unlabeled set. On iteration i the algorithm labels $U - N \times i$ inputs and trains with $L + N \times i$, each step takes a time of $O(D)$. Thus, each iteration takes $O\left(D(U - N \times i + L + N \times i)\right) = O\left(D(U + L)\right)$. There are about U/N iterations, and thus the total running time is $O\left(DU(U + L)/N\right)$, which scales quadratically with the number of unlabeled examples U. We reduced the time in two ways, first setting N to be proportional to U (e.g. $N = U/1000$), and second, using parallelization in the search for the set A_N^i.

5. Data

We downloaded a few million Amazon reviews of products from 35 categories, or domains. Beside the actual text, each review is additionally associated with a numeric ranking of 1 to 5 stars. We consider reviews with 4 or 5 stars as positive reviews, and reviews with 1

1. The bootstrapping procedure can be stopped at any time, in which case the last trained classifier would be returned (e.g., after a certain period of time, or when the best score is lower than some threshold).

Algorithm 1 Bootstrapping Framework

Input: labeled data set S_l, unlabeled data set S_u
Parameters:
- N number of inputs labeled on each iteration
- r parameter to be used by AROW
- n number of online iterations of AROW

Initialize: $i = 0$, $S_l^0 = S_l$, $S_u^0 = S_u$
while $|S_u^i| > 0$ **do**
- Receive a model using AROW
  $\boldsymbol{w}^i = AROW(S_l^i, r, n)$
- Select $A_N^i \subseteq S_u^i$ the N unlabeled instances with the highest score,
  $\min_{\boldsymbol{x} \in A_N^i} \boldsymbol{w}^i \cdot \boldsymbol{x} \geq \max_{\boldsymbol{x} \in S_u^i / A_N^i} \boldsymbol{w}^i \cdot \boldsymbol{x}$
- Update sets:
  $S_l^{i+1} \leftarrow S_l^i \cup \left\{ \left(\boldsymbol{x}, \text{sign}\left(\boldsymbol{w}^i \cdot \boldsymbol{x} \right) \right) \ : \ \boldsymbol{x} \in A_N^i \right\}$
  $S_u^{i+1} \leftarrow S_u^i / A_N^i$
- Set $i \leftarrow i + 1$

end while
Return the final classifier, $\boldsymbol{w}^{i-1}$

or 2 stars as negative. Reviews with 3 stars (about 9%) were shown in preliminary tests to be very hard to predict (or noisy) and thus were omitted. More than three quarters of the reviews in Amazon are positive, yet for development purpose, we made the dataset balanced, ending with 2.3 million reviews of each label, i.e., $4.6M$ reviews altogether. The total number of words–tokens separated by white spaces–is $700M$. All experiments below were repeated 5 times, each with a random draw of test set and labeled (training) set. Table 5 in the appendix (Haimovitch et al., 2012) summarizes the properties of the data.

Reviews were preprocessed as followed: (1) Convert upper-case text to lower-case. (2) Replace common non-word patterns (such as common emoticons, 3 dots, links) with a unique mark. (3) Remove HTML tags and any character that is neither alphanumeric nor a punctuation. (4) Expend abbreviations ("prof." → "professor"), and remove periods from abbreviations ("i.e." → "ie").

Additionally, since negations may inverse the sentiment of other words, we used the following processing of negation words: (1) Expand common apostrophe omissions (eg "don't" → "do not"), and (2) Replace (up to 7) words following a negation word (no, not, never, nobody) with a unique negation form. For example, the text "not so good" is replaced with "not neg_so neg_good". Finally, stop-words are removed.

Reviews are represented as a binary bag-of-word vector. We used both unigrams and bigrams, ending with about $33M$ features. We normalized the vector-reviews to have a unit Euclidean norm. The average number of non-zero elements over the 4.6M reviews is 192, with 99% of the reviews having at most $1,000$ non-zero elements, the shortest review has only 1 non-zero element, and the longest review has $5,922$ non-zero elements.

6. Empirical Study of SSL

The goal of these experiments is to evaluate the performance of the SSL algorithm on a single domain. We used the 8 domains with largest amount of examples: books, movies,

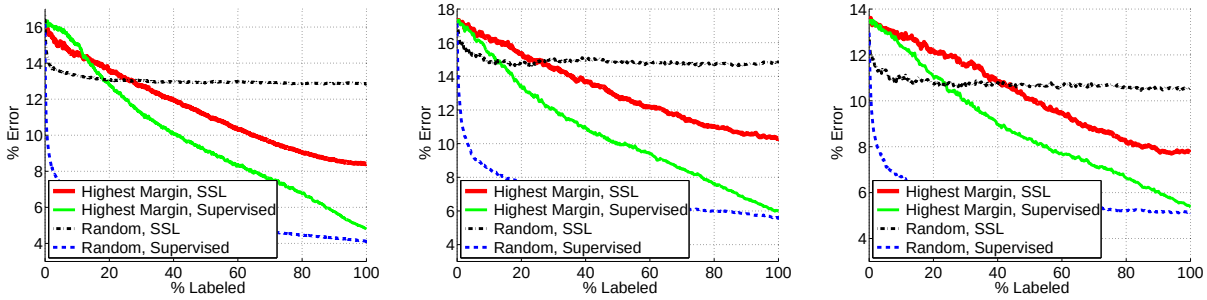

Figure 1: SSL results for three domains (left to right): Books, Movies and Electronics.

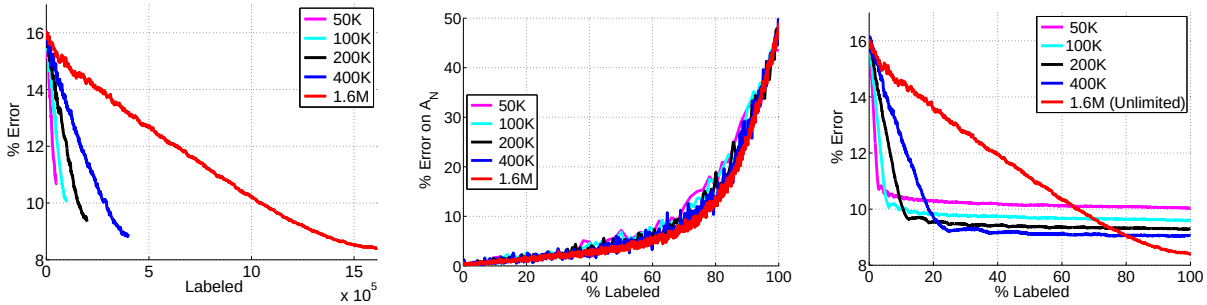

Figure 2: left: SSL Books - Different amounts of unlabeled data. middle: Error on added instances (1.6K each iteration). right: SSL in chunks.

	$\lvert S_u \rvert$	**Before**	**After**	**Skyline**
Domain	**No. Unlabeled**	**Label 1,000**	**SSL**	**Label all**
Books	$1.6M$	16.0%	8.4%	4.0%
Movies	$0.5M$	17.1%	10.3%	5.5%
Elect.	$0.4M$	13.4%	7.8%	5.1%
Music	$0.3M$	17.8%	9.8%	6.1%
Kindle	$0.2M$	16.0%	8.7%	6.1%
Videos	$0.1M$	17.3%	10.5%	7.3%
Kitchen	$0.1M$	13.7%	8.2%	5.5%
Health	$0.1M$	15.9%	10.1%	6.1%

Table 1: Test error of three algorithms on a single domain. Before: training with small amount of labeled data, after: semi-supervised learning, and skyline: training with all data labeled.

electronics, music, kindle, videos, kitchen, and health. From each dataset we randomly picked $1K$ examples with their labels to be the initial set S_l, and additional $10K$ examples for evaluation, or test-set, except the book domain for which we used $100K$ (it contains much more examples than other domains). The remaining examples consists of the initial unlabeled set S_u for each domain. We ran the bootstrapping algorithm for about $1K$

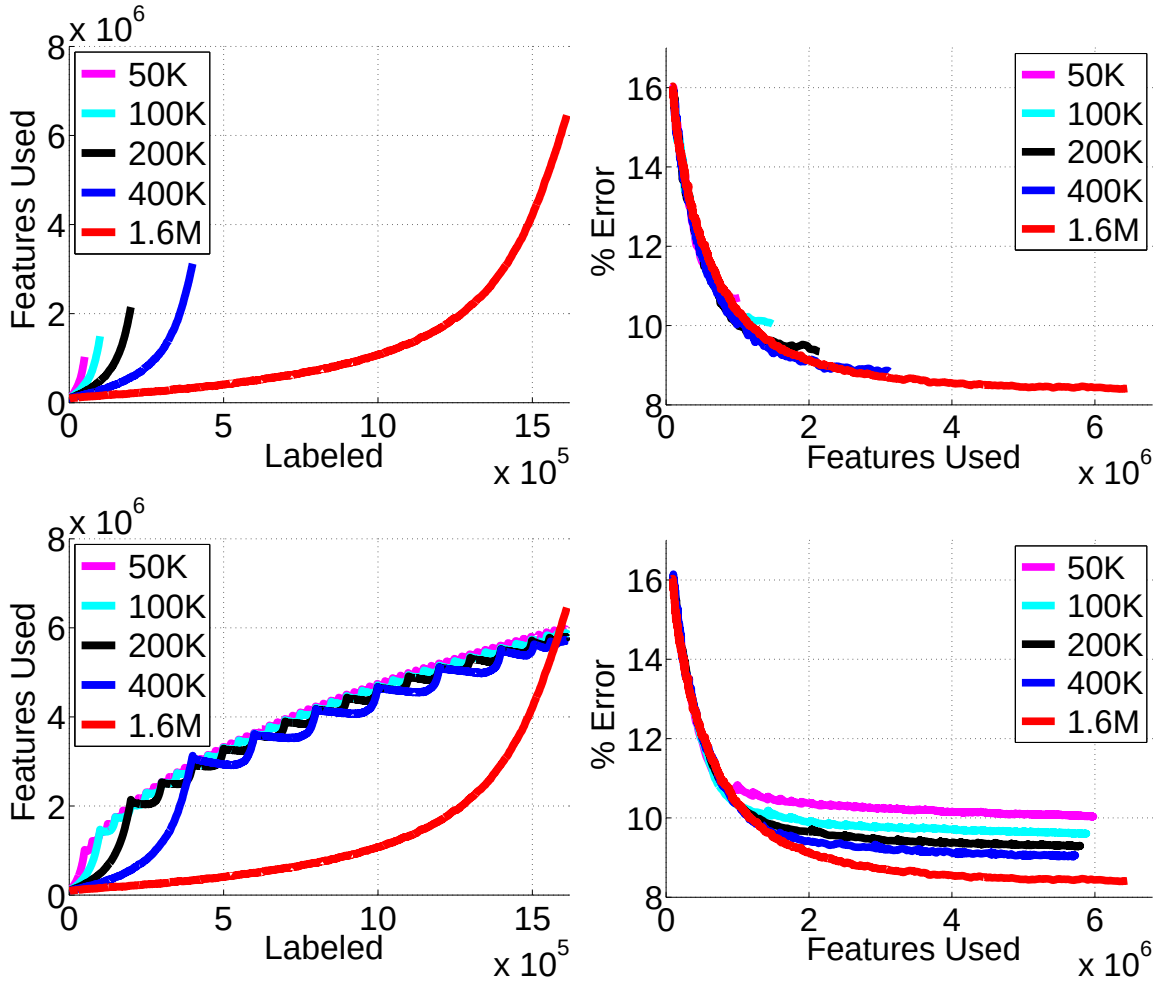

Figure 3: Top, left-to-right: No of features used by the model vs no labeled examples, test error vs. no of features; both panels apply to learning with a fixed size of unlabeled data. The next two panels (bottom) show the same plots but when adding unlabeled data in chunks.

iterations, that is we set, $N = |S_u|/1000$, and computed the error rate on the test set after each iteration.

The two left columns of Table 1 state the size of the unlabeled set for the eight datasets. The three right columns summarize the test error of three algorithms. Our baseline, called *before*, is training with only $1,000$ labeled examples. Next, we evaluate our bootstrapping algorithm, called *after*, and also evaluated a *skyline* version, trained with the entire set of examples labeled, that is, adding labels to the original unlabeled set S_u. For example, in the books domain we have $1,614K$ labeled examples, all together. For all eight domains, test error of SSL (*after*) is more than 40% lower than the test error of the baseline. Additionally, the algorithm was able to close at least 60% of the gap in test error between training with $1M$ examples and training with all the data.

Fig. 1 shows detailed (averaged) results for three domains. Our goal is to evaluate the contribution of each of the two components of the bootstrapping algorithm: choosing examples and labeling them. The first algorithm presented is Alg. 1, denoted by *Highest Margin, SSL* (red solid). The next algorithm, denoted by *Highest Margin, Supervised* (green solid), is a possible skyline, where instead of using $\boldsymbol{w}^i$ to label the new set A^i_N, the true labels are used. The difference between the performance of these two baselines illustrates the amount of additional error suffered by the SSL algorithm that is not using the true labels, but only generated ones. The third algorithm, denoted by *Random, SSL* (dot-dashed black) evaluates the contribution of our method to choose examples. Here, the algorithm chooses random N examples for A^i_N and uses the current model $\boldsymbol{w}^i$ only to label them. Finally, we used a standard supervised algorithm, denoted by *Random, Supervised* (dashed blue), that chooses random examples and uses the true labels. The x-axis in all plots is the fraction of the initial unlabeled data that is labeled (either by the algorithm or true labels) and used to build the model, and the y-axis is the error on the test set.

In all datasets (results for Music are omitted due to lack of space) the qualitative behavior is similar, and thus we focus on the left panel which shows the results for books. At $x = 0\%$ all algorithms use the same initial $1,000$ labeled training data and thus suffer the same test error of 16% (first row, third column of Table 1). The value at $x = 100\%$ for *Highest Margin, SSL* (red line) shows the test-error of the SSL algorithm after training 8.4% (first row, fourth column of Table 1), and the value of 4.0% for *Random, Supervised* (blue-dashed line) shows the test-error of the algorithm training with the fully labeled data (first row, fifth column of Table 1).

First, not surprisingly, supervised algorithms outperform SSL algorithms (blue and green lines are below black and red for most of the range). Second, the quality of the SSL-labeling rule degrades in time, the gap between the red and green line gets larger as more unlabeled data is being labeled. We investigate this phenomena below. Third, comparing the two self-labeling (with $\boldsymbol{w}^i$) methods–highest margin (red solid) and random (dashed black)–we observe that random outperforms highest-margin only on early iterations, but then, highest-margin outperforms a random choice of examples when most amount of unlabeled data is used. This is because the error of $\boldsymbol{w}^i$ on a random set is higher than its error on a chosen set (highest-margin), and as a consequence, the algorithm adds examples with higher label noise to the training set.

Random vs. Choosing Examples: We investigated the tradeoff between using random examples and highest-margin examples with SSL. Here random examples were used for the initial $p\%$ of the unlabeled data, and the remaining $(100 - p)\%$ were chosen using highest-margin. The value of $p = 0\%$ is the same as *Highest margin, SSL* of the left panel of Fig. 1, and the value of 100% is the same as *Random, SSL* of that figure. The results are summarized in Fig. 11 in the appendix (Haimovitch et al., 2012). In a nutshell, after labeling about 30% of the unlabeled points., using no random examples $p = 0\%$ (*Highest margin*) outperforms all other choices $p > 0$. In fact even for $p = 5\%$ the trend of the test error was similar to the trend of *Random, SSL*, leading to the conclusion that the algorithm could not recover even from 5% of the unlabeled data that was labeled with noisy labels.

Similarly, we experimented with initially using highest margin examples and then switching over to using random examples. The results can be viewed in Fig. 13 in the appendix

$\|S_l\|$	Before	After	Std. Dev.
100	30.8%	15.2%	7.20%
1,000	16.0 %	8.4%	0.24%
10,000	10.2%	7.1%	0.11%

Table 2: test error vs size of initial training data on books

(Haimovitch et al., 2012). The results clearly show that if the SSL algorithm is stopped before the unlabeled dataset is exhausted, then it is preferable to switch to random selection shortly before stopping. This can give a quick boost to the success rate, as the error rate drops more than 1 percentage point.

Amount of Unlabeled Examples: Careful study of the error rate of *Highest-margin, SSL* in all three panels of Fig. 1 make it seem that the slope of all test-curves start to reduce after $x = 80\%$ of the unlabeled data was (self-)labeled, which may indicate that the there is a limit to the usage of unlabeled data. Since, 1.6M reviews are all the unlabeled data we have, we repeated the experiments with *less* amount of labeled data. The left panel of Fig. 2 shows the test-error of the learned classifier for six-subsets (one of which is all $1.6M$ reviews) of the unlabeled data for the books domain. We observe, that the phenomena that the slope of the curve reduces close to the end of the dataset, appears in all scales (or dataset sizes). In all curves the slope close to the end is lower than the slope in the beginning. Furthermore, the test error after labeling an entire subset of $50K$ examples ($\sim 10\%$) is much lower that the test error after picking $50K$ examples from $1.6M$ ($\sim 15\%$). We hypothesize that it is because, a random $50K$ examples is more similar to the test data, than $50K$ examples chosen by the *Highest-Margin* method, which is close to the initial $1,000$ labeled examples. Only when most of the unlabeled examples are labeled, the effective training set becomes close to the test set and the performance improves.

We gain deeper insights into this phenomena in the middle panel of Fig. 2, where we plot the error of $\boldsymbol{w}^i$ over the chosen new set A_N^i of size $U/1,000$ vs the fraction of labeled inputs (which is proportional to i). Clearly, for all sizes of unlabeled data, the error over the new set grows about linearly until $x \approx 78\%$ (where the error is $\sim 10\%$), then the error starts to increase sharply until the prediction is equivalent to random labeling (50% error rate) in the last 3% of the data. This may indicate, that unlabeled data, independent of their size, contain a fraction of examples that are hard to label, nevertheless, including them, even with a very noisy label, still improves the performance of SSL (as indicated by the fact that the error lines are continuing to decrease).

We next examine the possibility of using unlabeled data gradually. That is, first training with $50K$ examples, then using additional $50K$, and so on, until using the entire $1.6M$ book reviews. Our hope was to have an error-curve that "connects" all the end-points of the curves of the left panel of Fig. 2. That is, have a fast convergence as using $50K$ examples, and a final performance as using $1.6M$ documents. The results are summarized in the right panel of Fig. 2. All runs use $1.6M$ unlabeled reviews, yet in chunks. Each curve corresponds to a run where the algorithm first used q examples, then after these q examples were all labeled, q additional examples were made available, and so on. From the plot we observe that performance was improved during the first chunk of size q, after that the curves are

	Col id	1	2	3	4	5	6	7	8
	Labeled	Src	Src	Src	Both	Both	Both	Trgt	Trgt
	Unlabeled	-	Trgt	Both	-	Trgt	Both	-	Trgt
Source Domain	Target Domain	1&2 Before	1 After	2 After	3&4 Before	3 After	4 After	SSL Before	SSL After
Movies	Books	19.2%	9.4%	10.1%	17.0%	8.6%	8.6%	16.0%	8.4%
Elect.	Books	27.4%	14.6%	22.5%	19.5%	9.2%	12.9%	16.0%	8.4%
Music	Books	20.5%	13.9%	13.4%	17.1%	11.8%	11.1%	16.0%	8.4%
Books	Movies	19.2%	11.1%	10.5%	17.7%	10.6%	10.0%	17.1%	10.3%
Elect.	Movies	25.9%	18%	14.1%	19.0%	10.7%	11.0%	17.1%	10.3%
Music	Movies	19.2%	11.2%	11.5%	18.4%	10.8%	10.5%	17.1%	10.3%
Movies	Elect.	23.7%	8.7%	11.0%	14.3%	7.9%	8.2%	13.4%	7.8%
Books	Elect.	24.7%	8.6%	21.2%	14.6%	7.9%	8.4%	13.4%	7.8%
Music	Elect.	26.7%	8.5%	12.3%	14.3%	7.9%	7.9%	13.4%	7.8%
Books	Music	19.8%	12.1%	13.2%	17.1%	9.7%	13.0%	17.8%	9.8%
Movies	Music	21.4%	13.9%	19.3%	17.9%	9.7%	9.6%	17.8%	9.8%
Elect.	Music	26.2%	21.2%	26.5%	19.4%	10.1%	10.4%	17.8%	9.8%

Table 3: Performance of algorithm in domain-adaptation.

almost flat, indicating that any additional amount of unlabeled data is not useful to further reduce the test error.

We investigate this phenomena in Fig. 3. The left panel shows the size of the model (=total number of distinct features, the "dictionary") during training for each size of unlabeled set (left panel Fig. 2). When more data is used, the total number of features increases, both during training (lines are increasing) and with larger sets of unlabeled data (the highest point of each curve increases). This indicates that our bootstrapping algorithm "covers" possible features much slower than of a random sample of the same size. For example, with $50K$ unlabeled examples there are about $1M$ features (hight of magenta line), yet when the algorithm is run with $1.6M$ unlabeled points, after $50K$ only about $200K$ features are used. This point is further made clear in the second panel of Fig. 3, where we plot the test error vs size of model (combining both left panels of Fig. 2 and Fig. 3). During most of the training process, the test error is reduced as more features are accumulated, except in the end (where very noisy labels are introduced). The next two panels of Fig. 3 are analog to the first two panels, yet they correspond to adding data in chunks (as in the right panel of Fig. 2). Comparing the first and third panel, clearly adding unlabeled data in chunks causes features to be added much faster than with no chunking. In fact all chunk sizes shown in the third panel (other than $1.6M$ which is no chunking) behave similarly. This rapid addition of features, as indicated by the right panel, coincides with worse performances per number of features used.

Label Noise: To test if the algorithm can work even in the case the initial training set is noisy, we conduct a few experiments where a random fraction of the labels were flipped. The results can be found in the appendix in Fig. 14 (Haimovitch et al., 2012). As would be expected, a noisy initial training set has lingering effects. However, even with 20% label noise, the behavior of the algorithm remains the same. For 1% and 5% the effect is negligible.

Unbalanced Data: We experimented with using the full data of the books domain, that is $6M$ reviews, where 85% of the data is positive. Since the initial $1K$ training set and test sets are chosen randomly, they too reflect the same ratio of positive vs. negative reviews. The results can be seen in Fig. 15 in the appendix (Haimovitch et al., 2012). In short, the behavior remains the same, where the error rate drops from 12.1% to 5.9%.

Substituting AROW: We compared the same SSL technique with 4 variants. The same bootstrapping framework was used, but with a different classifier: 1) Single epoch Perceptron, 2) Single epoch Averaged-Perceptron, 3) 10 epoch Perceptron, and 4) 10 epoch Averaged Perceptron. We used the balanced 1.6M book reviews for these experiments, starting with a $1K$ training set, as before. The results are summarized in Fig. 16 in the appendix (Haimovitch et al., 2012). The best result after AROW was for the 10 epoch Averaged Perceptron, which starts at 18.7% and ends at 13.6%. This means that both the initial error and the rate of improvement is inferior to using AROW. Interestingly, Perceptron after one epoch, outperforms its averaged version.

Comparison to T-SVM: We compared our method to the linear T-SVM implementation described and used by Sindhwani and Keerthi (2006). We chose this algorithm as it is well known, and similarly to ours, it is a generic SSL linear classifier intended for large scale data. For these experiments we used a subset of the balanced book reviews data: 1K reviews for the initial training set, 400K for the unlabeled set, and two 50K test sets. We compared our method with three methods: 1) L2-SVM-MFN, 2) multi-switch T-SVM, and 3) Deterministic Annealing semi-supervised SVM. The best results were found for the multi-switch T-SVM ($\lambda = 10^{-5}, \lambda_u = 10^{-3}, S = MAX$) ending with an average of 15.4%, compared to our 9.5%. It is also interesting to note that some of the runs took more than 24 hours, while ours consistently finishes within the hour. Further experiments are needed to verify and establish the cause of the discrepancy.

Amount of initial labeled data: We conclude this section with Table 2 that summarizes experiments studying the effect of the size of the initial labeled examples. In all runs we used $1,613K$ of unlabeled data. Thus any difference in performance is only due to difference in the amount of labeled data. We ran the algorithm with three sizes of that set, repeating the experiments 5 times. As in Table 1 two algorithms were evaluated, one using a small amount of labeled data (*before*), and one that uses SSL (*after*). As expected, more labeled data improve performance (error values are decreasing in the column *before*). Yet, SSL still improves performance. Even with initial $10K$ labeled examples, SSL is able to reduce test error by $\sim 30\%$. Also, the performance with $10K$ labeled examples is the same as with $1K$ examples and about $800K$ additional unlabeled examples (see Fig. 1).

7. Domain Adaptation

Motivated by the results of the last section, we employed our algorithm to the task of domain adaptation, where data from two domains are available. We are interested in the test error of only one of them, called the target domain. The SSL setting of the last section is where all data comes from the target domain. We evaluated four additional settings. (1) The initial labeled training set consists of a single domain (source domain) and the unlabeled set consists of the target domain(s), (2) Same as setting 1, except the unlabeled

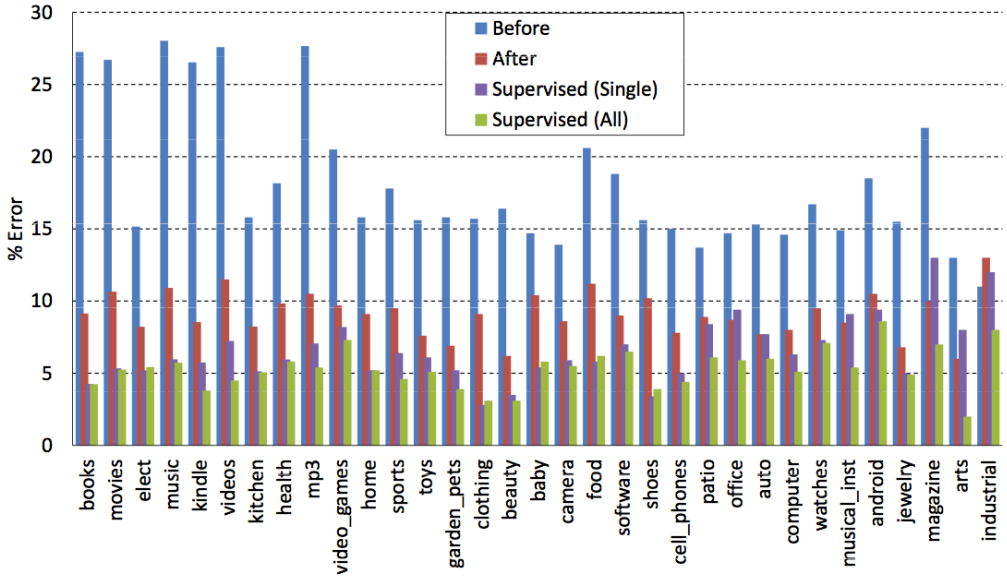

Figure 4: Domain adaptation from the industrial domain to 32 other domains. See text for details.

set also includes data from the source domain, (3) Same as setting 1, except the training set also includes data from the target domain, the total number of labeled data remains $1K$. (4) The initial training set and the unlabeled set both include data from the source and target domains, the total number of labeled data remains $1K$.

The results are summarized in Table 3 for twelve pairs of source and target domains. The numbers show the test error over the target domain. The first two columns (1,2) show the test error when training only with labeled source, or also when using unlabeled target data (setting 1). The next column (3) shows the results if we use unlabeled data from both domains (we omitted the results *before* as they are the same as of setting 1). The next two columns (4,5) show the results for setting 3, where we have small amount of labels from both domains, and unlabeled data only from the target domain. The next column (6) shows the results for when the initial training set and the unlabeled set come from both domains. Finally, we added in the last two columns (7,8) for completeness the results from Table 1 where there are only target data.

First, comparing the results with only labeled data (cols 1,4,7), we observe, as expected, that the more data we have from the target domain, the lower the test error is. For example, when the target domain is books, the test error is 19.2% when the labeled data is only movies, 17.0% if we replace half of the movies labeled data with books labeled data, and reduced to 16.0% when all labeled data is of books.

Second, comparing cols 1 and 2, we see that using in addition domain adaptation reduces error. For example (row 1) training with $1K$ labeled reviews from movies, the classifier achieves 19.2% test error on the books domain. Yet, using additional unlabeled books examples, it is able to reduce the error to 9.4%, which is lower than a test error of 10.2% obtained by building a classifier even with $10,000$ labeled books examples (see Table 2).

Encouraged from these results we investigated three possible alternatives to improve performance: use unlabeled data from both source and target domains (setting 2), use data from both domains during training (same amount altogether, setting 3), or use data from both domains for both the initial training set and the unlabeled set (setting 4).

Comparing columns 1 and 3 we conclude that when using unlabeled data from both domains (setting 2) our algorithm improves performance over just using the labeled data, yet comparing columns 2 and 3, we see that in 9 out of 12 cases, using unlabeled data only from the target domains yield lower test error than using unlabeled data from both source and target.

Finally, comparing columns 2, 5 and 7 we see that using 500 examples from each domain is a sweet point of having all data from only the source or only the target. For example, when the target domain is books, having $1K$ labeled examples from books yields a test error of 8.4% and when starting with $1K$ movies reviews the test error is larger by 1%, and is 9.4%. Similarly when the target domain is movies and starting with labeled movies reviews the test error is 10.3% and when having $1K$ labeled books reviews the test error is higher by 0.8% and is 11.1%. When starting with equal number of reviews from both domains, the test error on books is 8.6% (0.2% larger than labeling only book reviews) and on movies is 10.6% (0.3% larger than labeling only movies reviews).

We also performed a one-to-many adaptation starting with $1K$ labeled examples from the Industrial domain and using unlabeled data from *all domains*. We evaluated two other skylines that use more labeled data, one building a *single* classifier per domain, using all data from that domain, and the other, building one classier using *all* data. We expect the first baseline to perform well, since there is no bias due to examples from other domains, and the second baseline to perform well since the amount of training data is large (low variance).

The results are summarized in Fig. 4 with domains ordered by the amount of unlabeled data (books with largest amount). For all domains, except Industrial, the adaptation approach improves performance significantly over using only the labeled data, with an average relative reduction in test error of $47 \pm 15\%$. Both skylines are indeed better than the domain-adaptation algorithm, yet for domains with small amount of data the gap is not large. As was observed by Dredze et al. (2010) in a smaller scale, there is no clear winner between the two skylines.

In a similar experiment, we ran a one-to-many adaptation starting with $1K$ labeled examples from the Industrial domain and using unlabeled data from all domains, using *unbalanced data*. In this setting, 84.7% of the data used had positive labels, and the size of the unlabeled dataset used was $15M$. The results are similar to the balanced data experiments, with an average relative reduction in test error of $46 \pm 8\%$.

8. Weakly supervised learning (WSL)

Our bootstrapping algorithm is based on an initial set of labeled examples. We now describe a variant in which we replace this human effort with a way to generate such a labeled set automatically, yielding an unsupervised method.

The only deviation from the SSL algorithm is the construction and labeling of the initial set. Once it is generated the same bootstrapping algorithm is used. Instead of sampling a

Domain	$\|S_l\|$	$\|S_u\|$	Before	After
Books	$22.4K$	$1.6M$	24.1%	13.5%
Movies	$9.5K$	$0.5M$	24.2%	15.0%
Elect.	$9.3K$	$0.4M$	21.4%	11.2%
Music	$4.2K$	$0.3M$	25.9%	16.1%

Table 4: Size of initially labeled set with WSL rules, test error with model built using the initial set (before) and after applying the bootstrapping algorithm (after).

set of examples and label them by humans we used the following rules to pick documents which are likely to be positive or negative. We are aiming for a high-precision subset, otherwise the bootstrapping algorithm would fail, as seen above. Clearly, these rules are based on prior knowledge about the task at hand.

A review is considered positive if the following two conditions holds, one for the title, and one for both title and body. The title contains at most 2 words, where at least one of them belongs to the set (great, excellent, perfect, good, recommended). Additionally, the title and body, do not contain any negation word (no, not, never, nobody), nor a negative word (poor, awful, horrible, bad, disappointment).

Similarly, a review is considered negative if its title contains at most 2 words, where at least one of them is either a negative word, or a negated word of one of the five positive words above. Additionally, the title and body do not contain a positive word. Finally, we chose an equal number of positive reviews and negative reviews. These rules have high-precision, close to 100% of the chosen reviews are labeled correctly. Example book reviews that are true and false positive and true and false negative appear in Fig. 9 in the appendix (Haimovitch et al., 2012).

The results of four single domain experiments are summarized in Table 4. The initial labeled set by the rules is between $4.2 - 22.4K$, which is about $1 - 2\%$ from the available examples per domain. To our surprise, the test error using this large amount of training data was higher than the test error using $1K$ labeled examples that are chosen randomly (compare with *before* column in Table 1).

Since the precision of the rules is very high, we hypothesize that this gap is because the set of inputs these rules chose are not representative and far from a random sample. Yet, when applying the SSL procedure after the rules the resulting test error (column *after* in Table 4) in all domains is lower than the test error of using $1K$ labeled examples (column *before* in Table 1).

9. Related work and Summary

In the last decade, large volume of work was published in the area of sentiment classification. However, only recent researches have begun to focus on larger-scale data sets. For example, Godbole et al. (2007) describe a system which uses score based information retrieval techniques for constructing a sentiment lexicon on a relatively large dataset; they track hundreds of thousands of news and blog entries over time.

Goldberg and Zhu (2006), as well as other later works, used a graph-based semi-supervised algorithm, where nodes represent documents, and edges weights represent the

similarity between documents. This approach allowed them perform only transduction, while for us transduction is a byproduct, and they experiment on a few thousand movie reviews only. Sindhwani and Melville (2008) used a bi-partite graph (word-document) to allow for semi-supervised classification. Their method was extended to allow also for induction and was evaluated on a few thousand review and blog posts. Dasgupta and Ng (2009) used SVM for active semi-supervised classification of a few thousand reviews, where the general approach of starting with "easy" reviews is similar to ours, although for us it is a consequence of our method, rather than a design choice.

Blitzer et al. (2007) and Tan and Wang (2011) used the structural correspondence learning (SCL) algorithm and weighted-SCL (respectively) for domain adaptation, both use pivot features to adapt to new domain specific features. They use a few thousand Amazon reviews.

The closest work to ours is of Glorot et al. (2011) who also predict sentiment from Amazon reviews. Yet, the amount of data they use is an order of magnitude smaller than ours, and their work focuses on domain adaptation, while we work additionally in semi-supervised and weakly-supervised settings.

Turney (2002) applied WSL techniques on hundreds of product (Epinions) reviews from multiple domains. Their core concept is similar to our WSL approach: use rules to choose and label an initial set of reviews, and then apply a self-training technique.

While our work shares some aspects with these works, we are not aware of any large-scale sentiment analysis study similar to ours. Most previously used datasets contain few thousand documents, with a total word count less than a few millions (often much less). Most, if not all, works mentioned in a recent survey on SSL (Zhu, 2005) evaluate their algorithms on much less data than we do.

Summary and Conclusions: We described a study showing the usefulness of large amounts of unlabeled data in various settings for sentiment classification of Amazon reviews. The study indicates that a large amount of data is useful, and it is not clear, what is the limit of the amount, if any. Additionally, the order and rate of using this data affects performance of the final classifier. We hypothesize that by proper incorporation of the data, we may be able to learn the sentiment associated with words, or features, that appear only in the unlabeled data, leading to improved generalization. The exact nature of this aspect is remained to be studied.

We currently expand our methods for unbalanced datasets, and improve them by incorporating the confidence information provided by AROW.

Acknowledgments

We greatly thank Fujitsu Ltd for their generous support of the work.

References

Xue Bai. Predicting consumer sentiments from online text. *Decis. Support Syst.*, 50(4): 732–742, March 2011. ISSN 0167-9236. doi: 10.1016/j.dss.2010.08.024. URL http://dx.doi.org/10.1016/j.dss.2010.08.024.

John Blitzer, Mark Dredze, and Fernando Pereira. Biographies, bollywood, boomboxes and blenders: Domain adaptation for sentiment classification. In *In ACL*, pages 187–205, 2007.

Long-Sheng Chen, Cheng-Hsiang Liu, and Hui-Ju Chiu. A neural network based approach for sentiment classification in the blogosphere. *J. Informetrics*, pages 313–322, 2011.

Koby Crammer, Alex Kulesza, and Mark Dredze. Adaptive regularization of weight vectors. In *Advances in Neural Information Processing Systems 22*, pages 414–422, 2009.

Sajib Dasgupta and Vincent Ng. Mine the easy, classify the hard: a semi-supervised approach to automatic sentiment classification. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2 - Volume 2*, ACL '09, pages 701–709, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics. ISBN 978-1-932432-46-6. URL http://dl.acm.org/citation.cfm?id=1690219.1690244.

Mark Dredze and Koby Crammer. Confidence-weighted linear classification. In *In ICML 2008: Proceedings of the 25th international conference on Machine learning*, pages 264–271. ACM, 2008.

Mark Dredze, Alex Kulesza, and Koby Crammer. Multi-domain learning by confidence-weighted parameter combination. *Machine Learning*, 79(1-2):123–149, 2010.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, Bellevue, USAl, June 2011. Omnipress.

Namrata Godbole, Manjunath Srinivasaiah, and Steven Skiena. Large-scale sentiment analysis for news and blogs. In *Proceedings of the International Conference on Weblogs and Social Media (ICWSM)*, 2007.

Andrew B. Goldberg and Xiaojin Zhu. Seeing stars when there aren't many stars: graph-based semi-supervised learning for sentiment categorization. In *Proceedings of the First Workshop on Graph Based Methods for Natural Language Processing*, TextGraphs-1, pages 45–52, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=1654758.1654769.

Yoav Haimovitch, Koby Crammer, and Shie Mannor. More is better: Large scale partially-supervised sentiment classification - appendix. 2012. URL http://arxiv.org/pdf/1209.6329v1.pdf.

Tetsuya Nasukawa and Jeonghee Yi. Sentiment analysis: capturing favorability using natural language processing. In *Proceedings of the 2nd international conference on Knowledge capture*, K-CAP '03, pages 70–77, New York, NY, USA, 2003. ACM. ISBN 1-58113-583-1. doi: 10.1145/945645.945658. URL http://doi.acm.org/10.1145/945645.945658.

Bo Pang and Lillian Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the ACL*, pages 115–124, 2005.

Bo Pang and Lillian Lee. Opinion mining and sentiment analysis. *Found. Trends Inf. Retr.*, 2:1–135, January 2008. ISSN 1554-0669. doi: 10.1561/1500000011. URL http://dl.acm.org/citation.cfm?id=1454711.1454712.

Rudy Prabowo and Mike Thelwall. Sentiment analysis: A combined approach. *J. Informetrics*, pages 143–157, 2009.

Vikas Sindhwani and S. Sathiya Keerthi. Large scale semi-supervised linear svms. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '06, pages 477–484, New York, NY, USA, 2006. ACM. ISBN 1-59593-369-7. doi: 10.1145/1148170.1148253. URL http://doi.acm.org/10.1145/1148170.1148253.

Vikas Sindhwani and Prem Melville. Document-word co-regularization for semi-supervised sentiment analysis. In *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, ICDM '08, pages 1025–1030, Washington, DC, USA, 2008. IEEE Computer Society. ISBN 978-0-7695-3502-9. doi: 10.1109/ICDM.2008.113. URL http://dx.doi.org/10.1109/ICDM.2008.113.

Songbo Tan and Yuefen Wang. Weighted scl model for adaptation of sentiment classification. *Expert Syst. Appl.*, 38(8):10524–10531, August 2011. ISSN 0957-4174. doi: 10.1016/j.eswa.2011.02.106. URL http://dx.doi.org/10.1016/j.eswa.2011.02.106.

Simon Tong and Daphne Koller. Support vector machine active learning with applications to text classification. *Journal of Machine Learning Research*, 2:45–66, 2001.

Peter D. Turney. Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 417–424, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073153. URL http://dx.doi.org/10.3115/1073083.1073153.

Xiaojin Zhu. Semi-Supervised Learning Literature Survey. Technical report, Computer Sciences, University of Wisconsin-Madison, 2005. URL http://www.cs.wisc.edu/~{}jerryzhu/pub/ssl_survey.pdf.