Large Scale Semi-Supervised Sentiment Analysis

Yoav Haimovitch
Large Scale Semi-Supervised Sentiment Analysis

Research Thesis

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer Science

Yoav Haimovitch

Submitted to the Senate of the Technion — Israel Institute of Technology
Tevet 5772 Haifa December 2012
In loving memory of my father, mentor and role model,
Dr. Yaacov Haimovitch (1951-2011)
The research thesis was done in the Department of Computer Science under the supervision of Prof. Koby Crammer and Prof. Shie Mannor from the Department of Electrical Engineering.

Acknowledgments

To my supervisors, Prof. Koby Crammer and Prof. Shie Mannor, it was a great privilege learning from the both of you. Thank you for your support during what was a very difficult time for me and my family. I also thank my colleague Maayan Harel Gal-On for her help.

To my dear family, thank you for all the love, support, and encouragement.

The generous financial support of the Technion and Fujitsu Ltd. is gratefully acknowledged.
Publications

Parts of this work were published in:

Contents

Abstract 1

Abbreviations and Notations 2

1 Introduction 3

2 Problem Overview 5
  2.1 Sentiment Classification of Reviews . . . . . . . . . . . . . . . 5
    2.1.1 Negation . . . . . . . . . . . . . . . . . . . . . . . . . 6
    2.1.2 Different Domains . . . . . . . . . . . . . . . . . . . . 7
    2.1.3 Off-topic Sentiment . . . . . . . . . . . . . . . . . . . . 7
    2.1.4 Internet Language . . . . . . . . . . . . . . . . . . . . . 7
  2.2 Large Scale . . . . . . . . . . . . . . . . . . . . . . . . . . . 7
  2.3 Semi-Supervised . . . . . . . . . . . . . . . . . . . . . . . . 8
  2.4 Tying it all together . . . . . . . . . . . . . . . . . . . . . . 8

3 Background 9
  3.1 Supervised Binary Classification . . . . . . . . . . . . . . . . 9
  3.2 AROW . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 10
  3.3 Semi-Supervised Learning (SSL) . . . . . . . . . . . . . . . . 11
  3.4 Domain Adaptation (DA) . . . . . . . . . . . . . . . . . . . . 11

4 Data 12
  4.1 Raw Data . . . . . . . . . . . . . . . . . . . . . . . . . . . . 12
  4.2 Preprocessing . . . . . . . . . . . . . . . . . . . . . . . . . . 13
  4.3 Data Representation . . . . . . . . . . . . . . . . . . . . . . 14
5 Bootstrapped AROW
5.1 Bootstrapping ........................................ 17
5.2 Why would this work? ................................. 19
5.3 Complexity ............................................ 20

6 Empirical Study ........................................ 21
6.1 Single Domain ........................................ 21
   6.1.1 Random vs. Choosing Examples ................. 24
   6.1.2 Amount of Unlabeled Examples ................. 25
   6.1.3 Amount of initial labeled data ................. 28
   6.1.4 Substituting AROW ............................... 29
   6.1.5 Label Noise ...................................... 30
   6.1.6 Comparison to T-SVM ............................ 30
   6.1.7 Alternative Scoring Methods ..................... 31
   6.1.8 Unbalanced Data ................................ 34
   6.1.9 Number of iterations ............................ 34
6.2 Domain Adaptation (DA) ............................ 35
   6.2.1 One-to-Many ................................... 40
6.3 Weakly Supervised Learning (WSL) ................ 46
   6.3.1 Active Learning ................................. 48

7 Development ........................................... 50
7.1 Development and Resources ......................... 50
7.2 Preliminary Work, Side Tracks, and Failed Attempts .... 51
7.3 Parallelization and Distribution .................... 52
   7.3.1 Distributed bootstrapping ....................... 53
   7.3.2 Training on the new data and combining ........ 53
   7.3.3 Learning directly from the new data .......... 54

8 Related Work ........................................... 55

9 Summary and Conclusions ............................. 57
9.1 Summary ............................................. 57
9.2 Future Work ......................................... 58
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>Amazon Review</td>
<td>13</td>
</tr>
<tr>
<td>6.1</td>
<td>Semi supervised results for four domains</td>
<td>23</td>
</tr>
<tr>
<td>6.2</td>
<td>Switching between highest-margin and random selection</td>
<td>24</td>
</tr>
<tr>
<td>6.3</td>
<td>SSL (Books) - using different amounts of unlabeled data</td>
<td>25</td>
</tr>
<tr>
<td>6.4</td>
<td>SSL (Books) - Error on added instances</td>
<td>26</td>
</tr>
<tr>
<td>6.5</td>
<td>No of features used by the model vs no labeled examples</td>
<td>27</td>
</tr>
<tr>
<td>6.6</td>
<td>Test error vs. no of features</td>
<td>27</td>
</tr>
<tr>
<td>6.7</td>
<td>Bootstrapping with AROW vs. Perceptron</td>
<td>29</td>
</tr>
<tr>
<td>6.8</td>
<td>Initial label noise</td>
<td>30</td>
</tr>
<tr>
<td>6.9</td>
<td>Applying the new scoring function (1.6M Book Reviews)</td>
<td>32</td>
</tr>
<tr>
<td>6.10</td>
<td>SSL on 6M book reviews (85% positive)</td>
<td>33</td>
</tr>
<tr>
<td>6.11</td>
<td>SSL (1.6M Books) - number of iterations</td>
<td>34</td>
</tr>
<tr>
<td>6.12</td>
<td>Domain Adaptation, 1 to 1</td>
<td>39</td>
</tr>
<tr>
<td>6.13</td>
<td>DA from ‘Industrial’ to 32 other domains, balanced data</td>
<td>42</td>
</tr>
<tr>
<td>6.14</td>
<td>DA from ‘MP3’ to 32 other domains, balanced data</td>
<td>43</td>
</tr>
<tr>
<td>6.15</td>
<td>DA from ‘Movies’ to 32 other domains, balanced data</td>
<td>44</td>
</tr>
<tr>
<td>6.16</td>
<td>DA from ‘Industrial’ to 32 other domains, full data</td>
<td>45</td>
</tr>
<tr>
<td>6.17</td>
<td>Examples of reviews labeled by the WSL rules. Left to right:</td>
<td>46</td>
</tr>
<tr>
<td></td>
<td>true positive, false positive, true negative and false negative.</td>
<td></td>
</tr>
<tr>
<td>6.18</td>
<td>Weakly supervised results for four domains</td>
<td>47</td>
</tr>
</tbody>
</table>
Abstract

We describe a bootstrapping algorithm able to learn from partially labeled data. We report the results of an empirical study for using this algorithm 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 amounts of data, and in all cases a significant improvement over the baseline was shown.

The extensive empirical study includes a comparison to T-SVM (showing our method to be superior), an examination of various parameters and settings, and experiments such as a large-scale, 1-to-many domain adaptation, showing the potential usefulness of the described method. We show that the algorithm, which is an extension of AROW (Adaptive Regularization of Weight Vectors) to the semi-supervised setting, retains the relative superior effectiveness of AROW when applied to sentiment classification. In the weakly supervised setting, we show an extension of our method that allows it to begin with no labeled data, and using rules designed with prior knowledge, to automatically label an initial training set and thus use the bootstrapping method.

In addition, we discuss theoretical and practical scalability issues, and suggest potentially interesting directions for future research.
Abbreviations and Notations

AROW — Adaptive Regularization of Weight Vectors
SSL — Semi-Supervised Learning
WSL — Weakly-Supervised Learning
SVM — Support Vector Machine
T−SVM — Transductive Support Vector Machine
F — number of features in the entire dataset
\(w\) — weight vector, \(w \in \mathbb{R}^F\)
\(x\) — instance vector, \(x \in \{0, 1\}^F\)
\(S_l\) — labeled set
\(S_u\) — unlabeled set
\(S_l^i\) — labeled set in iteration \(i\)
\(S_u^i\) — unlabeled set in iteration \(i\)
\(N\) — number of examples to be selected and labeled each iteration
\(I\) — number of iterations
\(A_N^i\) — the \(N\)-sized subset of \(S_u\), selected to be labeled in iteration \(i\)
\(L\) — size of the initial labeled set, \(|S_l^0|\)
\(U\) — size of the initial unlabeled set, \(|S_u^0|\)
Chapter 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.

The three typical approaches to the problem of sentiment analysis are machine learning (ML), natural language processing (NLP), and information retrieval (IR) (Pang and Lee, 2008). While the three approaches are not mutually exclusive and are usually mixed, it is natural that researchers come from or lean more heavily towards one of these fields. We approached the problem from the machine learning world view.

We describe a bootstrapping algorithm and apply it in a semi-supervised
learning setting, where only a relatively 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, weakly supervised learning that replaces labeled documents with a set of rules designed with prior knowledge. We show that in all problems our algorithm is improving performance using this large amount of unlabeled data.

In the next chapter (2), we will discuss the different aspects of the problem. In chapter 3, we will give a technical background of the relevant machine learning methods in the context of this work. In chapter 4, we will discuss the data and features used. Chapter 5 will give formal and intuitive descriptions of the algorithm. The main section of this work, the empirical study, will be covered in chapter 6. Next we will discuss the development process (chapter 7), followed by a comparison to related work (chapter 8), and finally ending with the summary and conclusions in chapter 9.
Chapter 2

Problem Overview

In this chapter we will discuss the different challenges that comprise the problem we wish to address. We will begin by explaining the task of sentiment classification and the associated difficulties. We will then view the task in light of large scale data and semi-supervised learning, and tie all the aspects together.

2.1 Sentiment Classification of Reviews

The problem we address is sentiment classification of product reviews. Given a product review, we want to differentiate between an overall positive review and a negative one. We assume all (or nearly all) reviews express some sort of sentiment, and are either positive or negative. For a human, this task would usually seem easy, if not trivial. However, once the scale of the problem is considered (see the next section), it quickly becomes impractical for a human.

Sentiment classification is sometimes further differentiated into polarity and subjectivity classification. Polarity classification means differentiating between positive and negative sentiment, whereas subjectivity classification means differentiating between sentiment and non-sentiment (or opinion vs. fact). As will be discussed in chapter 4, for our data it is reasonable to assume all reviews contain sentiment, and more importantly, the polarity is practically given per review (in the form of the star rating) - therefore, we focus on sentiment polarity. In addition, we are specifically concerned
with binary polarity classification (i.e. is this review positive, yes/no?). In the context of this work, when we discuss sentiment classification we always refer to binary polarity classification, unless specifically noted otherwise.

In combination with other tools, sentiment classification can be built on for further analysis, such as what features of a product draw the positive/negative sentiment, how one product compares to another, and so on. Generally speaking, sentiment analysis can be very beneficial for producers/advertisers and consumers alike (in the case of product reviews). In certain domains, such analysis could also benefit politicians, activists, diplomats, journalists, or anybody else concerned with the opinions of the masses.

Understanding the sentiment of a given text is not only a worthy challenge but in the general case, it is also a very difficult one, computation-wise. It may require not only lexical, but also syntactic and even semantic processing. In addition to those challenging tasks, it may also require various levels of common sense and domain specific knowledge. Compared to other text analysis tasks (such as topic classification), sentiment analysis, and specifically polarity classification of reviews, has some unique problems. Examples of such problems would be the sensitivity to negation and to the differences between the various domains (Tang et al., 2009). Another problem, specific to product reviews, is the expression of sentiment towards subjects other than the product.

2.1.1 Negation

Consider a short review simply stating “this product is good” vs. “this product is NOT good”. Clearly, the addition of the word “not” flips the polarity from positive to negative. This is a unique problem to sentiment (polarity) classification. While there are other types of “polarity shifters” (Wilson et al., 2005), negation covers the majority of them, and is considered to be a key problem in sentiment classification (Na et al., 2004). Of course, the negation word does not necessarily come immediately before the sentiment word, for example: “Does not look very good” or “No one thinks that this is any good”. Rarer forms of negation may have a different structure, such as when the negation appears at the end of the sentence, e.g. “I used to think this was good, but now I don’t”, or even in another sentence.
2.1.2 Different Domains

While many machine learning problems are sensitive to the difference in domains, it is worthwhile to note what this means for sentiment classification. Obviously, there are some generic sentiment words and phrases, such as “excellent” or “not good”. However, some domain specific words might change their polarity, for example the word “unpredictable” would usually be positive in a book or movie review, but would be negative in an electronic device. This is one of the reasons why we would expect a classifier trained on one domain to be less accurate on another.

2.1.3 Off-topic Sentiment

In product reviews, sentiment may be directed at other topics, instead of the reviewed product, and this might confuse the classifier. Other topics covered in a review may be other products, specific product parts, the site or delivery system, and plot elements (in books and movies).

2.1.4 Internet Language

The language used in product reviews is not formal, to say the least. Grammar and spelling mistakes, intertwining of foreign languages (non-English), the use of non-alphanumeric symbols, and the abundance of slang and lingo, are but a few of the characteristics of free form reviews. These issues detract the efficiency of classic NLP (natural language processing) tools.

2.2 Large Scale

Large scale datasets pose some challenging problems, regardless of the domain. First, the solution is limited to scalable algorithms. This may involve a relatively complicated design and implementation, including multithreaded and distributed computation. An efficient time complexity is a necessary condition for scalability, but in the end it is the actual running time that is important (i.e. constants matter).

Development under these conditions, where a single experiment can take hours or even days to complete, let alone a batch of related experiments, is also more challenging than small scale problems.
Efficiency in memory is also important when considering a dataset of unlimited size, which puts even more constraints on the potential solution. In addition, any sort of task that requires a manual pass over the data, such as making sure the labeled data is labeled correctly, or removing badly formed or unwanted data, is practically impossible.

2.3 Semi-Supervised

We are interested in a setting where labeled data is relatively scarce, i.e. for the examples we are given, only for a small fraction of them we are also told whether they are positive or negative reviews. This is important since if we can show our method works for such a setting, it might be possible to use it for many other forms of data. For instance, one might use a similar approach to sum up free form surveys, or to detect and classify sentiment in Tweet, blogs, comments on Internet videos or articles (or even in the articles themselves). For these real world data, explicit sentiment polarity is not readily found (unlike the star rating of Amazon reviews).

Semi-Supervised Learning (SSL) is a harder problem than fully supervised learning, and is usually expected to achieve a lower success rate. SSL is further discussed in section 3.3. Lastly, it is worth noting that we are concerned only with inductive learning, i.e. all given data is used for training, and accuracy is measured on an unseen test set.

2.4 Tying it all together

Given a large number of unlabeled product reviews, and relatively few labeled reviews, we want to produce a sentiment classifier which will be able to classify new data with high accuracy. In short, we want to solve large scale semi-supervised sentiment classification.
Chapter 3

Background

This chapter is intended to give more formal definitions, and the technical background needed to understand the following chapters. All definitions and explanations are given in the context of this work, and so should be helpful even to those who are already familiar with the main concepts.

3.1 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 $x \in \{0, 1\}^F$ where for our data $F \approx 33M$ is the number of features (see chapter 4), 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 $\text{sign}(w \cdot x)$ for some weight vector $w \in \mathbb{R}^F$.

We assume the existence of a supervised algorithm that given a labeled set $S_l = \{(x_i, y_i)\}$ outputs a model $w$ that performs well on that set. That is, the fraction of reviews from $S_l$ for which $w$ 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.
3.2 AROW

The AROW algorithm is based on the CW (confidence weighted) framework (Dredze and Crammer, 2008). It is incremental, or online, and works in rounds. Similarly to many other online classifiers, AROW maintains a weight vector, which geometrically is a (linear) separating hyperplane. Often, it is initialized with the zero weight vector \( w = 0 \in \mathbb{R}^F \). On each round AROW picks an example from the labeled set \((x, y) \in S_l\), and uses it to update its current model \( w \). The algorithm maintains not only a weight vector \( w \in \mathbb{R}^F \) but also a (diagonal) matrix \( \Sigma \in \mathbb{R}^{F \times F} \), which together represent the mean and covariance of a Gaussian distribution. For practical reasons, i.e. memory constraints, we must use the diagonal matrix variant. The algorithm updates both weight-vector and covariance after processing any example.

The covariance gives a level of confidence per feature - the lower the variance, the more we can be sure of the mean value that was found. This gives us a different learning rate per feature, which enables the algorithm to gradually reduce the magnitude of the updates for features it has seen many times. This ability is the key reason as to why AROW works well on sentiment classification. To get a sense of why this is true, consider the following example of a negative product review: “This movie is amazing, as it makes staring at paint dry seem like fun...”. As the word “amazing” usually appears in positive reviews, we can assume this feature has both high (positive) weight and high confidence (low variance). This means that if and when AROW makes an update on this example, the weight will only slightly decrease. Thus, the feature “amazing” will remain associated with positive reviews, and AROW successfully prevents this feature from being ruined because of one outlier. Since many words (and word pairs) have a similar problem (a relatively uncommon usage which flips their sentiment polarity), AROW’s robustness to this issue is important when dealing with sentiment classification.

Two parameters controls the behavior of the AROW algorithm when it is executed in a batch setting: a general learning rate \( r \) and the number of rounds the algorithm iterates over the training set. We denote by \( w = AROW(S) \) 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 converges on \( r = 10^{-5} \) across many settings, with very similar results to a fixed value (a fixed value saves running time).

### 3.3 Semi-Supervised Learning (SSL)

In semi-supervised learning, algorithms are introduced not only to a labeled set of examples \( S_l \), but also to an additional set of unlabeled examples, denoted \( S_u \). The new set contains only input vectors (or feature vectors) with no labels. The goal of the learning algorithm is to build a classifier \( w \) 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.

As will be discussed in the next chapter, we use fully labeled data for our experiments, which allows precise error measurements. To simulate a real world setting where labels are scarce, we simply ignore the labels of the instances designated to be “unlabeled”.

### 3.4 Domain Adaptation (DA)

In this setting, we start out with labeled data from one domain (source), and we are required to be tested on data from a different domain (target). In the context of product reviews, one can think of a setting where, for example, labeled book reviews are given, but the test set is composed of movie reviews.

A trivial approach would be to simply train on the given training set, producing a classifier as usual, and use this classifier on the target domain(s). However, because of the differences between domains (see 2.1.2), this will usually lead to worse results than that of training on the target domain. Instead, we wish to adapt the classifier, either while training or after, to the target domain.
Chapter 4

Data

In this chapter we will present the data we used in this work. Following the properties of the raw data, we will explain the preprocessing and the final representation used in the experiments that will be described in chapter 6.

4.1 Raw Data

We downloaded over 16 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 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 purposes, 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 $700M$ (over $2,000M$ for the full dataset).

Table 4.1 and Table 4.2 (end of this chapter) show the number of reviews per domain for the balanced and full data respectively. Note that the reviews were downloaded up until July 2011, and there may be variations in the names of the categories between then and now.

Fig. 4.1 shows a typical Amazon review. In this example, we’d use a positive label (note the 5 stars), the title (“Worth every penny”) and the text (“I’ve scanned”...).
4.2 Preprocessing

Reviews were preprocessed as follows: (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) Expand abbreviations (“prof.” → “professor”), and remove periods from abbreviations (“i.e.” → “ie”).

Additionally, since negations may invert the sentiment of other words, we used the following processing of negation words: (1) Expand common apostrophe omissions (e.g. “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”. This is similar to previously used methods (Das and Chen, 2001).

This negation handling works fairly well, despite being somewhat coarse. One reason is that it covers the most common negations. Another reason is that while failing to negate a sentiment word can easily ruin the classification, negating a non-sentiment word (regardless of whether it was supposed to be negated or not) bears little difference.

In the preliminary fine tuning of the negation handling (using fully supervised AROW), negating the 7 words following the negation word (or up to the end of the sentence) was found to give the best results (negating words appearing before the negation word was also tested). The average
improvement is a little over 2 percentage points in terms of test error (from over 18% to a little over 16%). In addition, the standard deviation of the test error was reduced by a little over 1.8 percentage points (from over 2% to under 0.2%).

Finally, stop-words are removed (such as “the”, “a”, etc.), while retaining common yet sentimental words. Attempts at decreasing the number of features, namely stemming, did not seem to help (in terms of test error) and thus were not included.

4.3 Data Representation

Reviews are represented as a binary bag-of-words (BoW) vector. We used both unigrams and bigrams (word pairs), ending with about 33M features (over 66M for the unbalanced data). We normalized the vector-reviews to have a unit Euclidean norm.

The average number of non-zero elements per review 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. These statistics hold for both the balanced and unbalanced datasets.

It is worth noting that TF$\times$IDF (term frequency x inverse document frequency) representation does not work well for sentiment classification, especially if using a large number of documents to calculate the IDF. This is because many sentiment words are common (and thus get a much lower weight), whereas a lot of uncommon words (which receive a much higher weight) are simply noise, such as the character names in a book or a movie. TF$\times$IDF is useful in reducing the noise of stop words, but those can easily be removed using a fixed list.
<table>
<thead>
<tr>
<th>Domain</th>
<th># of Reviews</th>
<th>Test Set Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>books</td>
<td>1,713,900</td>
<td>100,000</td>
</tr>
<tr>
<td>movies (Movies &amp; TV)</td>
<td>555,208</td>
<td>10,000</td>
</tr>
<tr>
<td>elect (Electronics)</td>
<td>426,258</td>
<td>10,000</td>
</tr>
<tr>
<td>music</td>
<td>293,704</td>
<td>10,000</td>
</tr>
<tr>
<td>kindle (Kindle Store)</td>
<td>226,752</td>
<td>10,000</td>
</tr>
<tr>
<td>videos (Amazon Instant Videos)</td>
<td>138,228</td>
<td>10,000</td>
</tr>
<tr>
<td>kitchen (Kitchen &amp; Dining)</td>
<td>133,740</td>
<td>10,000</td>
</tr>
<tr>
<td>health (Health &amp; Personal Care)</td>
<td>107,336</td>
<td>10,000</td>
</tr>
<tr>
<td>mp3 (MP3 Downloads)</td>
<td>101,468</td>
<td>10,000</td>
</tr>
<tr>
<td>video games</td>
<td>88,360</td>
<td>1,000</td>
</tr>
<tr>
<td>home (Home Improvement)</td>
<td>77,656</td>
<td>1,000</td>
</tr>
<tr>
<td>sports (Sports &amp; Outdoors)</td>
<td>77,030</td>
<td>1,000</td>
</tr>
<tr>
<td>toys (Toys &amp; Games)</td>
<td>76,388</td>
<td>1,000</td>
</tr>
<tr>
<td>garden_pets (Home, Garden &amp; Pets)</td>
<td>72,542</td>
<td>1,000</td>
</tr>
<tr>
<td>clothing (Clothing &amp; Accessories)</td>
<td>65,860</td>
<td>1,000</td>
</tr>
<tr>
<td>beauty</td>
<td>58,156</td>
<td>1,000</td>
</tr>
<tr>
<td>baby</td>
<td>56,994</td>
<td>1,000</td>
</tr>
<tr>
<td>camera (Camera &amp; Photo)</td>
<td>46,228</td>
<td>1,000</td>
</tr>
<tr>
<td>food (Grocery &amp; Gourmet Food)</td>
<td>45,270</td>
<td>1,000</td>
</tr>
<tr>
<td>software</td>
<td>40,954</td>
<td>1,000</td>
</tr>
<tr>
<td>shoes</td>
<td>36,986</td>
<td>1,000</td>
</tr>
<tr>
<td>cell phones (Cell Phones &amp; Accessories)</td>
<td>36,100</td>
<td>1,000</td>
</tr>
<tr>
<td>patio (Patio, Lawn &amp; Garden)</td>
<td>33,194</td>
<td>1,000</td>
</tr>
<tr>
<td>office (Office Products)</td>
<td>17,958</td>
<td>1,000</td>
</tr>
<tr>
<td>auto (Automotive)</td>
<td>17,756</td>
<td>1,000</td>
</tr>
<tr>
<td>computer (Computer &amp; Accessories)</td>
<td>14,544</td>
<td>1,000</td>
</tr>
<tr>
<td>watches</td>
<td>12,960</td>
<td>1,000</td>
</tr>
<tr>
<td>musical_inst (Musical Instruments)</td>
<td>9,254</td>
<td>1,000</td>
</tr>
<tr>
<td>android (Appstore for Android)</td>
<td>7,452</td>
<td>1,000</td>
</tr>
<tr>
<td>jewelry</td>
<td>7,026</td>
<td>1,000</td>
</tr>
<tr>
<td>magazine (Magazine Subscriptions)</td>
<td>4,902</td>
<td>100</td>
</tr>
<tr>
<td>arts (Arts, Crafts &amp; Sewing)</td>
<td>4,084</td>
<td>100</td>
</tr>
<tr>
<td>industrial (Industrial &amp; Scientific)</td>
<td>1,142</td>
<td>100</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>4,605,454</strong></td>
<td></td>
</tr>
<tr>
<td>Domain</td>
<td># of Reviews</td>
<td>Test Set Size</td>
</tr>
<tr>
<td>-------------------------</td>
<td>--------------</td>
<td>---------------</td>
</tr>
<tr>
<td></td>
<td>(% Positive)</td>
<td></td>
</tr>
<tr>
<td>1. books</td>
<td>5,935,619 (85.56%)</td>
<td>100,000</td>
</tr>
<tr>
<td>2. movies (Movies &amp; TV)</td>
<td>1,722,415 (83.88%)</td>
<td>100,000</td>
</tr>
<tr>
<td>3. music</td>
<td>1,484,452 (90.11%)</td>
<td>100,000</td>
</tr>
<tr>
<td>4. elect (Electronics)</td>
<td>1,001,935 (78.73%)</td>
<td>100,000</td>
</tr>
<tr>
<td>5. kindle (Kindle Store)</td>
<td>772,286 (85.32%)</td>
<td>10,000</td>
</tr>
<tr>
<td>6. mp3 (MP3 Downloads)</td>
<td>490,857 (89.66%)</td>
<td>10,000</td>
</tr>
<tr>
<td>7. videos (Amazon Instant Videos)</td>
<td>374,834 (81.56%)</td>
<td>10,000</td>
</tr>
<tr>
<td>8. kitchen (Kitchen &amp; Dining)</td>
<td>353,840 (81.10%)</td>
<td>10,000</td>
</tr>
<tr>
<td>9. health (Health &amp; Personal Care)</td>
<td>289,042 (81.43%)</td>
<td>10,000</td>
</tr>
<tr>
<td>10. sports (Sports &amp; Outdoors)</td>
<td>243,011 (84.15%)</td>
<td>10,000</td>
</tr>
<tr>
<td>11. video_games</td>
<td>220,053 (79.92%)</td>
<td>10,000</td>
</tr>
<tr>
<td>12. toys (Toys &amp; Games)</td>
<td>217,264 (82.42%)</td>
<td>10,000</td>
</tr>
<tr>
<td>13. home (Home Improvement)</td>
<td>215,506 (81.98%)</td>
<td>10,000</td>
</tr>
<tr>
<td>14. clothing (Clothing &amp; Accessories)</td>
<td>213,033 (84.54%)</td>
<td>10,000</td>
</tr>
<tr>
<td>15. beauty</td>
<td>177,096 (83.58%)</td>
<td>10,000</td>
</tr>
<tr>
<td>16. garden_pets (Home, Garden &amp; Pets)</td>
<td>171,801 (78.89%)</td>
<td>10,000</td>
</tr>
<tr>
<td>17. camera (Camera &amp; Photo)</td>
<td>164,810 (85.98%)</td>
<td>10,000</td>
</tr>
<tr>
<td>18. baby</td>
<td>157,053 (81.86%)</td>
<td>10,000</td>
</tr>
<tr>
<td>19. food (Grocery &amp; Gourmet Food)</td>
<td>154,696 (85.37%)</td>
<td>10,000</td>
</tr>
<tr>
<td>20. shoes</td>
<td>149,614 (87.64%)</td>
<td>10,000</td>
</tr>
<tr>
<td>21. patio (Patio, Lawn &amp; Garden)</td>
<td>71,801 (76.88%)</td>
<td>1,000</td>
</tr>
<tr>
<td>22. cell_phones (Cell Phones &amp; Accessories)</td>
<td>67,092 (73.10%)</td>
<td>1,000</td>
</tr>
<tr>
<td>23. software</td>
<td>54,894 (62.70%)</td>
<td>1,000</td>
</tr>
<tr>
<td>24. office (Office Products)</td>
<td>54,699 (83.58%)</td>
<td>1,000</td>
</tr>
<tr>
<td>25. auto (Automotive)</td>
<td>48,501 (81.70%)</td>
<td>1,000</td>
</tr>
<tr>
<td>26. computer (Computer &amp; Accessories)</td>
<td>44,267 (83.57%)</td>
<td>1,000</td>
</tr>
<tr>
<td>27. watches</td>
<td>43,584 (85.13%)</td>
<td>1,000</td>
</tr>
<tr>
<td>28. musical_inst (Musical Instruments)</td>
<td>38,288 (87.92%)</td>
<td>1,000</td>
</tr>
<tr>
<td>29. jewelry</td>
<td>28,158 (87.52%)</td>
<td>1,000</td>
</tr>
<tr>
<td>30. arts (Arts, Crafts &amp; Sewing)</td>
<td>13,042 (84.34%)</td>
<td>1,000</td>
</tr>
<tr>
<td>31. android (Appstore for Android)</td>
<td>12,673 (70.60%)</td>
<td>1,000</td>
</tr>
<tr>
<td>32. magazine (Magazine Subscriptions)</td>
<td>10,584 (76.84%)</td>
<td>1,000</td>
</tr>
<tr>
<td>33. industrial (Industrial &amp; Scientific)</td>
<td>3,314 (82.77%)</td>
<td>100</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>15,000,114 (84.65%)</strong></td>
<td></td>
</tr>
</tbody>
</table>

Table 4.2: Properties of the full (unbalanced) data.
Chapter 5

Bootstrapped AROW

Here we describe the method we chose in order to take an online fully supervised classification algorithm (AROW), and apply it to a semi-supervised batch setting. Following the description and explanation of this method, we discuss the intuition as to why this method would be relevant for the problem at hand, and conclude with a complexity analysis.

5.1 Bootstrapping

We propose to use a bootstrapping approach for SSL. Our algorithm first builds a model $\mathbf{w}^0$ using the labeled data $S^0_l = S_l$, then using this model it picks a small subset $A^0_N$ of size $N$ from the unlabeled set $S^0_u$ and labels its inputs using the model $\mathbf{w}^0$, constructing a new labeled set,

$$S^1_l \leftarrow S^0_l \cup \{(x, y) : x \in A^0_N, \ y = \text{sign}(\mathbf{w}^0 \cdot x)\}.$$ 

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

The algorithm works in iterations. On the $i$th iteration, it uses AROW to build a model $\mathbf{w}^i$ based on the set $S^i_l$ of size $|S_l| + i \times N$, which is then used to choose and label a new set $A^i_N$ of size $N$ from the unlabeled set $S^i_u$ of size $S_u - i \times N$. This set $A^i_N$ is then labeled using $\mathbf{w}^i$ and moved from the unlabeled set $S^{i+1}_u$ to the labeled set $S^{i+1}_l$. The algorithm stops when the unlabeled set is exhausted, i.e., a round $i$ for which $S^i_u = \emptyset$.

Generally, the bootstrapping procedure can be stopped at any time, in
Algorithm 1 Bootstrapping Framework (using AROW)

**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:** $i = 0$, $S^0_l = S_l$, $S^0_u = S_u$

**while** $|S^i_u| > 0$ **do**
- Receive a model using AROW
  \[ w^i = AROW(S^i_l, r, n) \]
- Select $A^i_N \subseteq S^i_u$ the $N$ unlabeled instances with the highest score,
  \[ \min_{x \in A^i_N} |w^i \cdot x| \geq \max_{x \in S^i_u/A^i_N} |w^i \cdot x| \]
- Update sets:
  \[ S^{i+1}_l \leftarrow S^i_l \cup \{ (x, \text{sign}(w^i \cdot x)) : x \in A^i_N \} \]
  \[ S^{i+1}_u \leftarrow S^i_u/A^i_N \]
- Set $i \leftarrow i + 1$

**end while**

Return the final classifier, $w^{i-1}$

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). For completeness, in the next chapter we include 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 $w^i$ to assign a score $s(x)$ to each unlabeled example $x \in S^i_u$ 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 $x$ from the hyperplane defined by $w$ (the margin),

\[ s(x) = |w \cdot x| \propto |x \cdot w|/\|w\| \]

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

Finally, since AROW is an online algorithm, its output depends on the
order of the input examples. We found that the best tradeoff between speed, simplicity and diversity is to fix a random permutation over $S^i_0$ and to add the new set of examples $A^i_N$, labeled by the current model, before $S^i_1$. In other words, in the next round $i + 1$, AROW first learns with the recently added examples $A^i_N$ and only then it learns with remaining labeled examples.

The algorithm is summarized in Alg. 1.

5.2 Why would this work?

For intuition as to why the bootstrapping method we described above should be able to learn from both the labeled and unlabeled data (SSL), let us consider a qualitative interpretation.

As the classifier is first trained on the given labeled data, it behaves exactly like the supervised version. Let us assume the classifier has learned that the words “excellent” and “amazing” are a good indication that a given review is positive.

Next, we use this classifier to label and assign scores to all of the unlabeled instances. Let us assume that the higher the score, the more confident we can be of the label given by the classifier, and vise versa. For the purpose of this example, we can assume that when we find reviews which contain the words “excellent” and “amazing”, we can label them as positive with high confidence, thus they’ll receive a high score and will be selected to be added to the labeled set.

By adding them to the labeled set, the new classifier has a chance to learn new features. For instance, let us assume a review contains the words we already know (“excellent”, “amazing”, plus several common yet non-sentimental words), and the word “magnificent”, which we have yet to see. Thus, by labeling this review as positive, and adding it to the new training set, we can see how the new classifier could learn the sentiment of a word it had never seen before (in a labeled example). As the algorithm continues iteratively, it is now hypothetically possible for it to classify a new instance, which contains only new sentiment features (such as “magnificent”).

As for the scoring function (i.e. the method we use to assign a score to an unlabeled example), using the margin to rank the unlabeled examples makes sense for any linear model. This is true because we assume that the initial (or current) classifier is not completely wrong, or very different from
the true (or best) separating hyper-plane (at least on the dimensions it is non-zero). Thus, even if we should adjust the hyper-plane in any direction, in the general case, the farthest instances are less likely to be labeled differently, and so we can be more confident in their current classification. For the instances that are closest to the hyper-plane, not only does a small divergence from the true (best) hyper-plane can cause them to be misclassified, but even small levels of noise (which are common) can push them from one side to the other, flipping their label.

5.3 Complexity

We conclude this chapter with a complexity analysis of the algorithm.

Let $D$ be the maximum number of non-zero elements of any input example, $D = \max_x \|x\|_0$. We assume $D \ll F$, where $F$ is the total number of features (see 4.3). 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 $O(D(U - N \times i + L + N \times i)) = O(D(U + L))$.

There are about $U/N$ iterations, and thus the total running time is $O(DU(U + L)/N)$, which scales quadratically with the number of unlabeled examples $U$.

We reduced the time in two ways. First, by setting $N$ to be proportional to $U$ (e.g. $N = U/1000$), giving linear time in the number of total examples (i.e. $U + L$). Secondly, we further reduce running time by using parallelization in the search for the set $A^i_N$. For further discussion of the system and speed-up issues, see 7.3.
Chapter 6

Empirical Study

The empirical study is the main part of this work. We divide the different experiments into three main settings: single domain, domain adaption, and weakly supervised learning. All experiments below, unless noted otherwise, were repeated 5 times, each with a random draw of test set and labeled (training) set. The size of the test set for each experiment is determined by the domain (see tables 4.1 and 4.2). Instances do not repeat between the different test sets.

6.1 Single Domain

The goal of these experiments is to evaluate the performance of the SSL algorithm on a single domain. We used the 8 domains with the largest amount of examples: books, movies, electronics, music, kindle, videos, kitchen, and health. From each dataset we randomly picked 1K examples with their labels to be the initial set $S_i$, 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 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 6.1 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
Table 6.1: 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.

| Domain | $|S_u|$ | Before Label 1,000 | After SSL | Skyline 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%             |

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. 6.1 shows detailed (averaged) results for four 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 $w^i$ 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 $w^i$ 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 the qualitative behavior is similar, and thus we focus on the top left panel (Fig. 6.1a) 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 6.1). 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 6.1), and the value of 4.0\% for Random, Supervised (blue-dashed line) shows the

Figure 6.1: Semi supervised results for four domains
6.1.1 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 \textit{Highest margin, SSL} of Fig. 6.1a, and the value of $100\%$ is the same as \textit{Random, SSL} of that figure.
The results are summarized in Fig. 6.2a. 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. 6.2b. 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.

6.1.2 Amount of Unlabeled Examples

Careful study of the error rate of Highest-margin, SSL in all four panels of Fig. 6.1 makes 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. Fig. 6.3a shows the test-error of the learned classifier for five 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 than 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 Fig. 6.4, where we plot the error of \( \mathbf{w}' \) 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 Fig. 6.3a. That is, have a fast convergence as using 50K examples, and a final performance as using 1.6M documents. The results are summarized in Fig. 6.3b. 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 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. 6.5a. The figure shows the size of the model (=total number of distinct features, the “dictionary”) during training for each size of unlabeled set (Fig. 6.3a). 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 Fig. 6.6a, where we plot the test error vs size of model (combining Fig. 6.3a and Fig. 6.5a). 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). Fig. 6.5b and Fig. 6.6b are analog to Fig. 6.5a and Fig. 6.6a, yet they correspond to adding data in chunks (as was shown in Fig. 6.3b). Comparing the Fig. 6.5a and Fig. 6.5b, clearly adding unlabeled data in chunks causes features to be added much faster than with no chunking. In fact all chunk sizes shown in Fig. 6.5b (other than 1.6M which is no chunking) behave similarly. This rapid addition of features, as indicated by Fig. 6.6b, coincides with worse performances per number of features used.

### 6.1.3 Amount of initial labeled data

Table 6.2 summarizes experiments studying the effect of the size of the initial labeled examples. In all runs we used 1,613K of unlabeled data.

| $|S_t|$ | 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 6.2: Test error vs. size of initial training data on books
(book reviews). 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 6.1 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 \( \sim 30\% \).

In addition, it is worth noting that the performance of a fully supervised AROW with 10K labeled examples is similar to that of the SSL algorithm with only 1K labeled examples and about 800K additional unlabeled examples (see Fig. 6.1).

### 6.1.4 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. 6.7. 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.

### 6.1.5 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 Fig. 6.8. 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.

### 6.1.6 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. It is also analogous to our extension of AROW
to the semi-supervised setting, as T-SVM is a similar extension of SVM.

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 (the results below are an average of the error on the two sets).

We compared our method with three different SVM methods: 1) L2-SVM-MFN (fully supervised), 2) multi-switch T-SVM, and 3) Deterministic Annealing semi-supervised SVM.

The fully supervised SVM uses only the initial 1,000 labeled examples, and the best result was 15.8% average test error, comparable to the 16% achieved by a single epoch AROW (see Table 6.1, books, before). The best results were found for the multi-switch T-SVM ($\lambda = 10^{-5}, \lambda_u = 10^{-1}, S = \text{MAX}$) ending with an average of 14.8%, 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. The time discrepancy is no surprise, as the worst case time complexity of T-SVM is $O((L + 2U)^3)$, where it is claimed (Sindhwani and Keerthi, 2006) that it typically behaves as only quadratic. In comparison, the bootstrapped AROW is linear in $L + U$, and always behaves the same. It is also important to note that these results are when we assume the three parameters of T-SVM are fine-tuned, something that takes additional time and is not always easy to do in a semi-supervised setting.

The difference in the reduction of error could be partially attributed to the qualities of AROW (see 3.2), namely the advantages in sentiment/text classification. As can be seen when AROW is replaced with another online algorithm (see 6.1.4), we can hypothesize that some of the reduction in error is due to the online nature of the classifier used. By using an online classifier, and by adding the newly labeled instances to the beginning of the training set, we force the new classifiers to use new features, thus allowing for better SSL.

6.1.7 Alternative Scoring Methods

For most of our SSL experiments we used the highest-margin scoring method. In one set of experiments, we attempted to enhance this method using the covariance matrix $\Sigma$ maintained by AROW (see 3.2). For a given instance
where $\alpha \in \mathbb{R}$ is a parameter. While the first part of the function is the margin, the second and new part is an estimation of how confident AROW is in the weights used. Since lower values of $\Sigma$ mean a higher confidence, a negative $\alpha$ will bias the selection towards confidence, and a positive $\alpha$ will bias the selection towards uncertainty, or novelty. The magnitude of $\alpha$ determines the relative weight of this confidence measure vs. the margin (setting $\alpha$ to zero gives only the margin as before).

Using this method we can get an improvement in the decline of test error, but only a slight improvement in the final test error. The results of the set of experiments (5 splits) is shown in Fig. 6.9.

We used the 1.6M book reviews, and tested the scoring function with $\alpha$ ranging from −6 to 6 (integer values only); for clarity, not all values are shown. The best results were found for $\alpha = -2$ (a little over 0.2 percentage points better than the baseline), however the purpose of these experiments was not to fine tune this parameter, but rather to assess the potential of
this approach (i.e. exploiting the covariance matrix).

First, the red line is the baseline of using the highest-margin alone ($\alpha = 0$). It is clear from the results that a positive $\alpha$, that is a bias toward novelty (less known features), is strictly worse than the baseline – in the graph, only $\alpha = 2$ is shown (black dashed line), but the results are very similar (and strictly worse) for larger values. For $\alpha < 0$, compare the red line and the blue line ($\alpha = -2$) – it shows that this method can improve the SSL.

The dashed magenta line, for $\alpha = -6$, shows a mixed result. On the one hand, it seems to indicate that a faster drop in test error could be achieved, at least in the first quarter. When allowed to run to 100%, it is worse than the baseline. There seems to be some potential for this scoring function, however it is not clear what guarantees (or expectations) could be made regarding the effect on the intermediate and final test error.

We have tried to exploit the covariance matrix in other similar ways, using multiplication (or division) instead of addition, but none were better than the baseline. For our purposes, the current findings were not good enough to be incorporated into the main algorithm, as this alternative requires more CPU time, and is difficult to fine tune.
6.1.8 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. 6.10. In short, the behavior remains the same, where the error rate drops from 12.1% to 5.9%.

6.1.9 Number of iterations

If we set the initial number of labeled data, it still remains to be decided how many iterations the bootstrapping should run. Setting the number of iterations as a constant parameter $I$, we can set the number of examples selected each iteration to $N = U/I$, where $U$ is the number of initially unlabeled examples.

Fig. 6.11 summarizes a 5 split set of experiments (std. deviation error bars). The initial number of labeled data is set at 1K, and the unlabeled data is the 1.6M book reviews, with test sets of 100K as before. The test error is the final error after all the unlabeled data has been labeled (i.e. after the last iteration).

It is interesting to note that even using only 100 iterations (i.e. $N \approx 16K$) comes close to the more expensive settings. The results lead to our de-
cision to use $I = 1K$, as it clearly preforms relatively well, and the algorithm seems robust to changes in this parameter.

6.2 Domain Adaptation (DA)

Motivated by the results of the single domain experiments, we employed our algorithm to the task of domain adaptation (see 3.4), 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 previous 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 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 $1K$. For all the settings, we set $N = 100$ and used the 5 split data as in the single domain experiments.

The results are summarized in Table 6.3 for twelve pairs of source and target domains (Fig. 6.12, appearing in the end of this subsection, shows the results for settings 1 and 3). 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 6.1 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 6.2). 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).
<table>
<thead>
<tr>
<th>Source Domain</th>
<th>Target Domain</th>
<th>Settings</th>
<th>Setting 1 &amp; 2 Before</th>
<th>Setting 1 After</th>
<th>Setting 2 Before</th>
<th>Setting 2 After</th>
<th>Setting 3 &amp; 4 Before</th>
<th>Setting 3 &amp; 4 After</th>
<th>SSL Before</th>
<th>SSL After</th>
</tr>
</thead>
<tbody>
<tr>
<td>Movies</td>
<td>Books</td>
<td>19.2%</td>
<td>9.4%</td>
<td>10.1%</td>
<td>17.0%</td>
<td>8.6%</td>
<td>8.6%</td>
<td>16.0%</td>
<td>8.4%</td>
<td></td>
</tr>
<tr>
<td>Elect.</td>
<td>Books</td>
<td>27.4%</td>
<td>14.6%</td>
<td>22.5%</td>
<td>19.5%</td>
<td>9.2%</td>
<td>12.9%</td>
<td>16.0%</td>
<td>8.4%</td>
<td></td>
</tr>
<tr>
<td>Music</td>
<td>Books</td>
<td>20.5%</td>
<td>13.9%</td>
<td>13.4%</td>
<td>17.1%</td>
<td>11.8%</td>
<td>11.1%</td>
<td>16.0%</td>
<td>8.4%</td>
<td></td>
</tr>
<tr>
<td>Books</td>
<td>Movies</td>
<td>19.2%</td>
<td>11.1%</td>
<td>10.5%</td>
<td>17.7%</td>
<td>10.6%</td>
<td>10.0%</td>
<td>17.1%</td>
<td>10.3%</td>
<td></td>
</tr>
<tr>
<td>Elect.</td>
<td>Movies</td>
<td>25.9%</td>
<td>18%</td>
<td>14.1%</td>
<td>19.0%</td>
<td>10.7%</td>
<td>11.0%</td>
<td>17.1%</td>
<td>10.3%</td>
<td></td>
</tr>
<tr>
<td>Music</td>
<td>Movies</td>
<td>19.2%</td>
<td>11.2%</td>
<td>11.5%</td>
<td>18.4%</td>
<td>10.8%</td>
<td>10.5%</td>
<td>17.1%</td>
<td>10.3%</td>
<td></td>
</tr>
<tr>
<td>Movies</td>
<td>Elect.</td>
<td>23.7%</td>
<td>8.7%</td>
<td>11.0%</td>
<td>14.3%</td>
<td>7.9%</td>
<td>8.2%</td>
<td>13.4%</td>
<td>7.8%</td>
<td></td>
</tr>
<tr>
<td>Books</td>
<td>Elect.</td>
<td>24.7%</td>
<td>8.6%</td>
<td>21.2%</td>
<td>14.6%</td>
<td>7.9%</td>
<td>8.4%</td>
<td>13.4%</td>
<td>7.8%</td>
<td></td>
</tr>
<tr>
<td>Music</td>
<td>Elect.</td>
<td>26.7%</td>
<td>8.5%</td>
<td>12.3%</td>
<td>14.3%</td>
<td>7.9%</td>
<td>7.9%</td>
<td>13.4%</td>
<td>7.8%</td>
<td></td>
</tr>
<tr>
<td>Books</td>
<td>Music</td>
<td>19.8%</td>
<td>12.1%</td>
<td>13.2%</td>
<td>17.1%</td>
<td>9.7%</td>
<td>13.0%</td>
<td>17.8%</td>
<td>9.8%</td>
<td></td>
</tr>
<tr>
<td>Movies</td>
<td>Music</td>
<td>21.4%</td>
<td>13.9%</td>
<td>19.3%</td>
<td>17.9%</td>
<td>9.7%</td>
<td>9.6%</td>
<td>17.8%</td>
<td>9.8%</td>
<td></td>
</tr>
<tr>
<td>Elect.</td>
<td>Music</td>
<td>26.2%</td>
<td>21.2%</td>
<td>26.5%</td>
<td>19.4%</td>
<td>10.1%</td>
<td>10.4%</td>
<td>17.8%</td>
<td>9.8%</td>
<td></td>
</tr>
</tbody>
</table>

Table 6.3: Performance of the algorithm in domain-adaptation.
(a) Movies to Books (Setting 1)  

(b) Movies&Books to Books (Setting 3)  

(c) Elect. to Books (Setting 1)  

(d) Elect.&Books to Books (Setting 3)  

(e) Books to Movies (Setting 1)  

(f) Books&Movies to Movies (Setting 3)
Figure 6.12: Domain Adaptation, 1 to 1
6.2.1 One-to-Many

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

Using the balanced data, we set $N = 100$. For the first experiment, the source domain was the ‘industrial’ domain (lab equipment, machinery, etc). The results are summarized in Fig. 6.13 with domains ordered by the amount of unlabeled data (‘books’ with the largest amount, ‘industrial’ with the least). For all domains, except ‘industrial’, the adaptation approach improves performance significantly over using only the initially labeled data, with an average relative reduction in per-domain test error of $49.4 \pm 9.5\%$ (source domain excluded). The total test error (combining all test sets into one), is reduced from $24.5\%$ down to $9.5\%$. Note that the initial classifier is at $11\%$ when tested on the source domain, where as the final classifier gets a nearly equal or lower error on all the other domains. The classification of the source domain in this case worsens, perhaps due to loss of domain-specific features. In any case, the adaption can be considered successful.

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. Next, we used ‘MP3’ as the source domain. The results are summarized in Fig. 6.14. Here the average relative reduction in per-domain test error is $47.7 \pm 8.5\%$ (source domain excluded), and the total test error, is reduced from $19.6\%$ down to $11\%$. When using ‘movies’ as the source domain (summarized in Fig. 6.15), the average relative reduction in per-domain test error is $54.9 \pm 9.6\%$ (source domain excluded), and the total test error is reduced from $18.6\%$ down to $10.1\%$.

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. We limited the number of iterations to 10K. The results are similar to the balanced data experiments, with an average relative reduction in test error of 46 ± 8% for each domain, and the total error is reduced from 14% down to 7.5%.
Figure 6.13: DA from ‘Industrial’ to 32 other domains, balanced data.
Figure 6.14: DA from ‘MP3’ to 32 other domains, balanced data.
Figure 6.15: DA from ‘Movies’ to 32 other domains, balanced data.
Figure 6.16: DA from ‘Industrial’ to 32 other domains, full data.
6.3 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 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.

<table>
<thead>
<tr>
<th>excellent story</th>
<th>great fiction</th>
<th>poor</th>
<th>It's horrible!!</th>
</tr>
</thead>
<tbody>
<tr>
<td>i usually get my books from the library but this one was so good i wanted to get my own copy!!! excellent story!</td>
<td>There's nothing really scientific on this book. It's only a great fiction work really. Almost comic most of the time. Have that in mind when you read it!</td>
<td>sorry Barbara i usually love your books but this one is really poor. the ending is a mess there is no character development no scene painting ... intreaging.</td>
<td>It is so horrible to believe that this truely happens to children and their parents. I wish I had read the book yrs. ago but thank the lord my children were okay.</td>
</tr>
</tbody>
</table>

Figure 6.17: Examples of reviews labeled by the WSL rules. Left to right: true positive, false positive, true negative and false negative.
close to 100% of the chosen reviews are labeled correctly. In fact, in some of cases where the label associated using Amazon stars disagree with the label assigned by the rules, manual inspection showed that the later is correct, and the original labeling is at fault.

Example book reviews that are true and false positive and true and false negative appear in Fig. 6.17 (with grammar and spelling mistakes intact). The results of four single domain experiments are shown in /figrefusl and summarized in Table 6.4. 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 6.1).
| 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 6.4: 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).

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 6.4) in all domains is lower than the test error of using 1K labeled examples (column before in Table 6.1).

It is also important to emphasize that while the rules are based on prior knowledge and are well thought out, the words they are looking for are very general, and represent rudimentary common knowledge, therefore a non-expert could surely generate a similar list very quickly. This, in comparison to the previous settings, where the user might be required to manually label at least 1,000 examples, clearly shows the motivation of this approach. If the lists of words are given to the user (which is possible because of their generality), then for the user the algorithm is completely unsupervised and requires no manual labor.

### 6.3.1 Active Learning

Since using the WSL method described above did not match the SSL version in terms of success, we experimented with a middle-ground approach, i.e. incorporate active learning. In this setting, the user first defines the WSL rules (or uses predetermined ones), just as before. However, during the course of the bootstrapping, the user will be presented with reviews the algorithm has a hard time classifying.

In our experiments, for each iteration of the algorithm, in addition to selecting the $N$ highest-margin instances, the lowest-margin instance is also
selected. This single instance is then given the true label (simulating a human classifier). The motivation for this is that a low margin instance has less known features. However, this approach did not work. Nevertheless, we believe that further investigation in this direction could be fruitful.
Chapter 7

Development

The development process was a challenge, mostly due to the size of the dataset. Here we describe the three development related issues. First, the resources used and the applications that were developed. Next, we describe some of the work that was done as part of the development. Lastly, we describe the different methods we attempted to employ for the sake of scalability.

7.1 Development and Resources

Two Perl scripts were developed for gathering the data from Amazon.com. The first script is a crawler that finds the links for the product reviews. The second script follows the links, downloads and extracts the relevant information. The download rate was slow as to not interfere with the website. The data was downloaded over a period of about six months. Both scripts were tailored for Amazon, and were tested thoroughly and tweaked as needed. Perl was chosen for the ease of parsing and the resilience to error (as the scripts were running 24/7). Finally, a few additional scripts were written for combining the downloaded data, and checking for errors.

The code for the classification program was written in Java 6. It has extensive test coverage, as some parts were written using Test Driven Development. It is very modular, supporting a vast amount of optional settings. The code was developed using variants of agile methodologies. Java was chosen for the ease of testing and agile development, the ease of implementing
design patterns, and for the ease of parallelization.

The experiments were executed on Odin: a 64 bit Linux cluster, with 33 nodes, each with at least 12 GB of memory and an 8 core Intel E5540 2.53GHz CPU. For the larger experiments (those with more than a million or two reviews), 10 designated high memory nodes were used, each having 48G of memory.

### 7.2 Preliminary Work, Side Tracks, and Failed Attempts

Aside from the different scoring methods we tried in 6.1.7, and the attempts at incorporating active learning in 6.3.1, there were other attempts that failed, or rather did not come to full fruition. First, in the preliminary work, we attempted to use simple (fast) methods to reduce the number of features, and more importantly, to standardize features. Stemming did not improve test error, and simple standardization techniques, such as replacing all “uo” with “ou” (a common mistake) or removing repeating letters, did not help much either. As we were more interested in the learning aspect, rather than the NLP or application, we were happy to not linger on these issues.

Perhaps the most significant direction we pursued (and abandoned) was an attempt at using the structure of the reviews. First, in a manual inspection of the reviews AROW classified incorrectly, we noticed that more than 80% of them could be avoided if only a certain part of the review was to be considered. Namely, the title, the first sentence, and the last sentence (as opposed to the basic title+body), all of which sometimes summarized the entire sentiment of the review, before (or after) going into detail – which is mostly noise for our classifier. We began by trying to build a meta-classifier, of which the inputs would be the output of four classifiers, trained and tested on one of the three parts mentioned before, or the entire text. We also tried to cross the training and the testing, giving the meta-classifier 16 inputs. When trained using lots of labeled data (we tried 50K), this approach showed an improvement over the baseline, of almost 2 percentage points. However, we were interested in settings were labeled data were scarce. Moreover, the overhead was too daunting to be considered for use with large scale data. This direction too was abandoned in favor of semi-
supervised learning. In addition for it not being scalable, it was less generic and assumed the structure of a product review (while we preferred a generic method).

Next, we tried to train a subjectivity classifier for titles, that is, detect when a title contains sentiment, in the hopes that in such cases we could ignore the body and easily classify the polarity of the review via the title alone. This did not work at all, for various reasons, mainly the lack of labeled data, and the need for better, or rather more specific, feature generation (such as detecting when the name of a product or creator is in the title). However, the knowledge gained here poised us well to design the rule based weakly-supervised learning method described in 6.3.

We had also done a few experiments which included the 3 star reviews, and we classified them as negative reviews (as defined by Amazon). Seeing as these reviews were noisy and caused many mistakes, we experimented with a classifier that could classify between the 3 star reviews and the rest. We envisioned a system where noisy reviews are removed before the main sentiment classification takes place. Generally speaking, 3 star reviews could be divided into two groups: those that are actually ambiguous, mixed, or lacking sentiment, and those that should have been given another star rating (typically a 2 or a 4, but sometimes even a 1 or a 5). For the SSL setting, we would like to get rid of the former, and keep the latter. However, in that point in time we have yet to have a functioning SSL algorithm, and so we decided to focus on making it work without the 3 star reviews. In the future, we can use the results of the fixed dataset as a skyline for the original (which requires the 3 star handling).

Lastly, as we were starting on the main research topic (large scale SSL), we experimented with graph based algorithms. However, we quickly abandoned them in favor of the bootstrapped AROW method, as they were hard to scale, much more volatile and dependent on NLP methods (e.g. smoothing), and much more in need of fine tuning - which is hard to do in a setting where most of the data is unlabeled.

### 7.3 Parallelization and Distribution

For practical reasons and for scalability, we attempted several ways to speed up the bootstrapping process. The process can be divided into two parts:
selecting new data, and retraining.

For selection of new data, we have two types of methods. The first is random selection, which is quite fast, as it does not require processing of the unlabeled data. The second and main method of selection involves assigning a score to the unlabeled data, then choosing the top $k$. It is easy to see that this is can be parallelized - the data can be split up into subgroups, and the top $k$ is found for each group; then, the instances that were selected are combined into a single group, achieving in theory and practice a near linear speedup.

Looking at the second part of the bootstrapping process, the retraining, we see the on-line classifier we use (i.e. AROW) now has to make a pass on all of the labeled data, in order. We experimented with a few different ways in order to reduce the run time.

### 7.3.1 Distributed bootstrapping

First, to reduce the total time, we experimented with the following distributed setting: the unlabeled data was split up into sub groups, divided among different machines, each running an instance of the program. Finally, we would combine the different results into one single classifier. This works fairly well. For example, in an experiment with 1.6M unlabeled data, selecting 100 new instances each iteration, it takes 13 hours to run the SSL process on a single machine (7 threads), and only 1 hour and 10 minutes when split to 4 machines. The combined classifier achieves a slightly higher error rate, which is expected, giving 8.7% compared to the single machine run which gives 8.4%. We tried combining the classifiers using different methods, but none were significantly different than the simple averaging of the weight of each feature.

### 7.3.2 Training on the new data and combining

We attempted to reduce retraining time by training a classifier on the new data, and then combining it with the current classifier, as opposed to training a new classifier on all labeled data (current+new). The combination was done similarly to the distributed version, however here no method was found to work.
7.3.3 Learning directly from the new data

We attempted to completely avoid the need for retraining by directly learning from the new data. As the instances with the highest margin are selected, this means that simply allowing the current classifier to train on these instances might not even lead to any updates, and those that will cause an update will not update the weights for the new features enough. We attempted various ways of stochastically choosing which features to include in the new instances, usually giving preference to features the classifier is the least confident about. We have not found a method that produces any desirable results.
Chapter 8

Related Work

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 adapta-
tion, while we work additionally in semi-supervised and weakly-supervised
settings.

Turney (2002) applied WSL techniques on hundreds of product (Epin-
ions) 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.
Chapter 9

Summary and Conclusions

To conclude this work, we will begin by summarizing the previous chapters, and end by discussing possible future work.

9.1 Summary

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 showed that our method is robust to noise and to changes in parameters and settings. This robustness is important, since fine tuning with few labeled examples is hard to do. Other desirable properties of our method are the consistently fast running time, with linear time complexity; the possibility of an any-time stop, as empirically the test error drops in a close to monotone fashion; and finally, the method is generic, and works for various domains and different settings.
9.2 Future Work

Future directions for this work may include incorporating the confidence information provided by AROW. Further work may focus on improving performance and scalability, and perhaps incorporating active learning coupled with WSL. One promising direction of increasing scalability is of course distributed computation, as the first step in that direction (see 7.3.1) is encouraging. In addition, it would be interesting to attempt to use this system for solving different problems (other than sentiment classification), and to further compare it to the alternatives.

As this work was meant to be generic, with a focus on machine learning (namely SSL), we refrained from using complex NLP methods, especially those that would use specific knowledge of the English language or be costly in resources. It is reasonable to assume that syntactic parsers or some sort of semantic analysis would increase accuracy, however, any such future development would need to consider the tradeoff of accuracy vs. scalability. Moreover, in a real world application, any increase in time complexity will likely reduce the amount of data that can be processed in a given time, thus by reducing the scalability, the accuracy could indirectly be reduced as well.

Last but not least, it would be useful to develop an online version of the algorithm. An online version could work on a stream of data, as opposed to a given dataset, which would be more applicable to real world problems.
Bibliography


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

נניח כי נתונות לנו: ניתן להבין מדוע אנו מצפים שתהליך זה יעבוד בצורה האינטואיטיבית הבאה: שבחרו באקראיות מתוך סך ספור דוגמאות מתויגות מספר מועט יחסית של דוגמאות לא מתויגות ומספר רב של "מומלץ". ניתן להניח שהמילים, אמן מסווג על הדוגמאות המתויגות ונקבל מסווג, כאמור. כל הדוגמאות. עתיענש שהמסווג למד לעשות את השיוך הזה, הרבה ביקורות חיוביות מופיעות ב"מדהים". נניח ששאר ל"מאורע" מופיעה ביקורת מכילה בין השאר את שתי המילים המתייחסות לדוגמאות שבקבוצת "מרהיב" המילה 'מרהיב' הובנו בעבר. ביקורת זו, המסווג למד do את הביקורת הזו כחיובית ברמת בטחון גבוהה, ביקורת姆ילה שנייה, היא מופיעה רק המילה 'מרהיב', לדוגמה. בהם. מקבלים מסווגים מדויקים יותר ויותר. המרחק של הדוגמאות, על מפריד-מכיוון שהמסווג בו אנו משתמשים פועל באמצעות מישור בשご利用 של תהליך המינוף השגיאה על קבוצת ו, מדד זה התברר כמוצלח. מהמישור משמש כמדד בטחון טבעי, חצי יותר מאף ב罩מסופים, באופן משמעותי הבדיקה צומצמה, הפעלנו אלגוריתם זה בתצורה של למידה מונחת למחצה, במערך ניסויים מקיף-astically, ולא מומלץ,wał, "מאורע" או "מדהים". וואריאציות שונות, פרמטרים, בדקנו מספר רב של תצורות, כמו כן, קבוצת, קבוצת האימון ההתחלתית. פעמים, בכל אחת מחמש הריצות (ללא חזרות) הוגרלו, וקבוצת הבדיקה, הדוגמאות הלא מתויגות-hard. מהpropertyNameaturals.伍 illustrate never. אנו דווקא מעוניינים בסיווג דוגמאות מתחום אחר, (ביקורות על ספרים, לדוגמה) להחליף תוצאות פחות טובות מאשר מסווג שאומן על "שימוש במסווג שאומן על תחום אחר בד".各式, ושתי הדרכים נחוצה בתצורה בה קבוצת האימון שייכת לתחום א-בין (או אדפטציה) התאמת, לדוגמה) אך אנו דווקא מעוניינים בסיווג דוגמאות מתחום אחר במגוון שיטות. התחום הדרוש, כלא נדע לחלק את "מכיוון שבמציאות בד.תחום אחד ליותר משלושים תחומים אחרים בו זמנית, או כמה, העובדה שהאלגוריתם לא דורש לדעת כמה תחומים קיימים, הדוגמאות הלא מתויגות לתחומים. תמצית את ההשערה שהוא מסוגל להיות שימושי בתנאיי אמ, דוגמאות יש מכל תחום. תצורה זו באה לפתור את. למידה מונחת באופן חלש היא למעשה קרוב למידה עצמאית לחלוטין, האלגוריתם מתחיל רק עם דוגמאות. כאן. הבעיה גם לתייג מספר קטן יחסית של דוגמאות עלול להיות יקר. הספר כלשהו יש צורך במכיתות(וEDIUM CONTINUE)כדי שהתהליך שתואר קודם יוכל להתחיל, אולם, לא מתויגות. האלגוריתם יקבל רשימה קצרה של מילים חיוביות (: החלשה) כאן נכנסת ההנחיה. של דוגמאות מתויגות, את אלא תמלינון באמצעותرشימה וחילופין. עבודה מועטה ביותר לעומת תיוג ידני של דוגמאות. המשתמש בתהליך הוא עצמאי לחלוטין, אנו דנים במשמעויות התיאורטיות והמעשיות של עיבוד כמות גדולה של מידע, בנוסף לניסויים. לתנאי הרי זה,📢�, הגדרות מסוימות האלגוריתם רץ בזמן ליניארי ביחס לכמות הדוגמאות. אנו דנים בשיטות ההנדסיות העיקריות בהן השתמשנו כדי, באופן מעשי, הכרחי לעבודה בקנה מידה גדול, (ליבות של המחשב/כומר ניצול כל המעבדים)השימוש שלנו בתכנות מקבילי, בפרט, להאיץ את התהליך. ההlogueحم הוא כמובן נחוץ. (על יותר ממחשב אחד במקביל עבודה)وهם הניסיונים על תחומים נוספים (בעוד במכיתות)(לדוגמה מ trabal neste צוות של חברות)либо 통해 בשיטות רבות, או בשיטות של חיפוש. מענה לשאלות של מחקר בדיקות (לא כמו)המשתמש בצורה לקויה, ראו. המטרות של עבודתו זו נותרו, שלוש פעמים שלו, אך התוצאות התוצאות של המש掣ת, ומופיעה. הצלחת, קיימא, ומטרו, אך纹ה وغيرها.
The goal of emotion analysis, in particular, is to understand the opinions and feelings of people.

This is important for predicting outcomes and, for example, in the tabular form, the emotions can be written in the text, so that it can be understood quickly, the way the data is presented is also used in public companies to discuss and evaluate their products.

One of the basic tasks of emotion analysis is sentiment classification, which means distinguishing between positive or negative emotions.

The set of all reviews of Amazon products (Amazon.com), similar research has been done on sentiment classification of product reviews. In contrast, most studies used a controlled experiment, we used a large amount of data, 15 million reviews, sorted by rating, the subject of the review, and words in the text. In a review, a word or phrase is a sentiment, and in the text that appears in the title and body of the review, we used stars for rating, for each review, it is a component that appears continuously in the text. Each word is a component. To create a component vector, a component is present in the review, then the number of components is normalized to 1.

This vector is very sparse, and the operation needed for sentiment classification is in the memory and vector multiplication, which is much easier to compute, but only for a small number of reviews. In this research, we focused on a structure with a lot of information, and we used sentiment classification for all the information.

Therefore, we ignored the classification for the rest of the information, because we believe that the classification of the rest is very important, especially without any restrictions, it is much easier to understand, because for many problems, we can get information that is not available.

We must remember that success of this research, in terms of time and money, is often very expensive, and information is often used in this structure. Therefore, we need a different approach, a supervised learning algorithm that depends on the amount of data given (or self-sufficient) we focused on supervised learning, which has a small amount of labeled data.

This extension, for supervised learning, a supervised learning algorithm exists for this purpose, to make it easier to interpret the data with labeled data, we can use an algorithm for supervised learning, and use it also for unlabeled data, so that when given a new classification, we can check if it is the best, and also when trained, we can use it to improve the point compared to the initial point.

Akpakar Shai Alonim, through this thesis, and the two authors thank the Technion - Computer Science Department - M.Sc. Thesis - MSC-2013-03 - 2013.
הודות

למניחים של, פרופ' קובי קרמר ומרפ', שי מנור, ויחד זו זכות גדולה, להנחיית המחשב למדעי בפקולטה וחברון בהנחייתה של夫人 הנדרית שיני מנור. בתודה על ההבנה והתמיכה בתוכניות compart הועברה על עלי על השמות, תודה רבה על תודתי עם דוקטורנים נוספים ומייני הרצל ו-אול.(transformed)

למשפחתי יקר, תודה על ההבהבה והתמיכה וה.userAgent.

אני מודה לתלמידי הפקולטה ולפילוגים בשתי מועדון על התמיכת célף הנדיבתי

בשטה מתויה.
מזכות לזכרו של אביו וומרה דוד, ד"ר יעקב חיותי (1951-1991)
ניתוח הולך והך בקנה מידה גדול
באמצעות למידה עצמאית-למחצה

יובו על מחקר

לעשות מילוי תחלק של הדרישות לקבלית תואר שני מגיסטר למדעי
במדעי המחשב

יותב חיימוביץ

הውש לוגט חטכני–מנון טכנולוגיה לישראל

דצמבר 2012 חמה סבת תשע”ב
ניתוח ההלך והרוח בקנה מידה גדול
באמצעות למידה עצמאית-למחצה

יואב חיימוביץ