We're Not in Kansas Anymore: Detecting Domain Changes in Streams

†⋆**Mark Dredze** and †⋆**Tim Oates** and †‡**Christine Piatko**
†Human Language Technology Center of Excellence,
⋆Center for Language and Speech Processing,
‡Applied Physics Lab
Johns Hopkins University
⋆University of Maryland, Baltimore County
mdredze@cs.jhu.edu, oates@umbc.edu, christine.piatko@jhuapl.edu

Abstract

Domain adaptation, the problem of adapting a natural language processing system trained in one domain to perform well in a different domain, has received significant attention. This paper addresses an important problem for deployed systems that has received little attention – detecting when such adaptation is needed by a system operating in the wild, i.e., performing classification over a stream of unlabeled examples. Our method uses $\mathcal{A}$-distance, a metric for detecting shifts in data streams, combined with classification margins to detect domain shifts. We empirically show effective domain shift detection on a variety of data sets and shift conditions.

1 Introduction

Consider a named entity recognition system trained on newswire stories. Given annotated documents containing sentences like "*Tony Hayward* has faced fresh criticism for taking time off to go sailing ..." we would like to learn a model that will allow us to recognize that "Obama" and "BP" are named entities in a sentence like "Obama summoned BP executives ...". When all of the documents come from one data distribution, like newswire articles, this tends to work well. However, the sentence "OBAMA SUMMONED BP EXECUTIVES ..." from transcribed broadcast news, and others like it, will probably lead to poor results because the features it relies on have changed. For example, capitalization patterns are no longer a good indicator of the presence of a named entity and appositives are not indicated by punctuation. This problem of *domain shift* is a pervasive problem in NLP in which any kind of model – a parser, a POS tagger, a sentiment classifier – is tested on data that do not match the training data.

Given a model and a stream of unlabeled instances, we are interested in automatically *detecting* changes in the feature distribution that negatively impact classification accuracy. For example, a sentiment classification model trained on book reviews may heavily weight n-grams features like "uplifting" and "page turner". Those features may never occur in reviews of kitchen appliances that get mixed in at test time, and useful features in this new domain like "efficient" and "noisy compressor" will have never been seen during training and therefore not be in the model. Furthermore, we do not assume labeled instances are available to help detect these harmful changes. Other tasks related to changes in data distributions, like detecting concept drift in which the labeling function changes, may require labeled instances, but that is not the focus of this paper.

There is significant work on the related problem of adapting a classifier for a known domain shift. Versions of this problem include adapting using only unlabeled target domain data (Blitzer et al., 2006; Blitzer et al., 2007; Jiang and Zhai, 2007), adapting using a limited amount of target domain labeled data (Daumé, 2007; Finkel and Manning, 2009), and learning across multiple domains simultaneously in an online setting (Dredze and Crammer, 2008b). However, in practical settings, we do not know *if* the data distribution will change, and certainly not when. Additionally, we will not know to *what* do-

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 585–595,
MIT, Massachusetts, USA, 9-11 October 2010.

main the shift will happen. A discussion forum devoted to science fiction books may change over time to focus more on fantasy and then narrow to discussions of vampire fiction. Maybe this shift is harmless and it is possible to identify the sentiment of the discussants with the original model with no loss in accuracy. If not, we seek methods that detect this shift and trigger the use of an adaptation method.

Our domain shift detection problem can be decomposed into two subproblems: detecting distributional changes in streams of real numbers, and representing a stream of examples as a stream of real numbers informative for distribution change detection. We select the $\mathcal{A}$-distance metric (Kifer et al., 2004) to solve the first subproblem since it has been previously used in other domain adaptation work (Blitzer et al., 2006; Blitzer et al., 2007). Our main contribution is towards the second problem, representing examples as real numbers for this task. We demonstrate that classification margins, which incorporate information about features that most impact system accuracy, can effectively solve the second subproblem. Furthermore, we show that the previously proposed Confidence Weighted learning algorithm (Dredze et al., 2008) can provide a more informative measure than a simple margin for this task. Our experiments include evaluations on commonly used domain adaptation data and false change scenarios, as well as comparisons to *supervised* detection methods that observe label values, or have knowledge of the target domain.

We begin with a description of our task and previous applications to language data. After describing the data used in this paper, we discuss the $\mathcal{A}$-distance metric and how it has previously been used for adaptation. We then show that margin based methods effectively capture information to detect domain shifts, and propose an alternate way of generating informative margin values. Finally, we compare our results to settings with supervised knowledge, and close with a survey of related work.

2 Domain Shifts in Language Data

The study of domain shifts in language data has been the purview of domain adaptation and transfer learning, which seek to adapt or transfer a model learned on one source domain with labeled data to another target domain with few or no labeled examples. Formally, errors from such transfers have two sources: differences in feature distributions and changes to labeling functions (annotation standards) (Ben-David et al., 2006; Ben-David et al., 2009). Empirical work on NLP domain shifts has focused on the former. For example, Blitzer et al. (2007) learned correspondences between features across domains and Jiang and Zhai (2007) weighted source domain examples by their similarity to the target distribution.

We continue in this tradition by making two assumptions about our setting. First, a change in domain will be signaled by a change in the feature distributions. That is, new words, phrases, syntactic structures, etc. signal that the system has shifted to a new domain. Second, while there may be a change in the labeling function, i.e., features have a different meaning in each domain, this will be a secondary concern. For example, both Daumé (2007) and Dredze and Crammer (2008b) assume that domains are more similar than different.

A similar problem to the one we consider is that of concept drift, where a stream of examples are labeled with a shifting labeling function (concept) (Nishida and Yamauchi, 2007; Widmer and Kubat, 1996). While concept drift is similar there are two important differences. First, concept drift can be measured using a stream of *labeled* examples, so system accuracy is directly measured. For example, Klinkenberg and Joachims (2000) detect concept drift with support vector machines, using estimates of leave-one-out performance to adaptively adjust and maintain a training window that minimizes estimated generalization error. This is possible only because class labels arrive with the examples in the stream. Another concept drift detection algorithm, STEPD, uses a statistical test to continually monitor the possibly changing stream, measuring system accuracy directly, again using the labels it receives for each example (Nishida, 2008). Obviously, no such labels are available in our unsupervised setting. Second, concept drift assumes only changes in the labeling function, whereas domain adaptation relies on feature distribution changes.

Several properties of detecting domain shifts in natural language streams distinguish it from traditional domain adaptation, concept drift, and other related tasks:

- **No Target Distribution Examples** Blitzer et al. (2007) estimate the loss in accuracy from domain shift by discriminating between two data distributions. In our setting, we have no knowledge of the target distribution.

- **No Labeled Target Data** Some approaches to domain adaptation assume a limited number of labeled examples (Daumé, 2007; Dredze and Crammer, 2008b; Finkel and Manning, 2009). We assume no labels in our setting.

- **Online Setting** Domain adaptation typically assumes a batch transfer between two domains. We consider a purely stream (online) setting.

- **Computationally Constrained** Our approach must be fast, as we expect to run our domain shift detector alongside a deployed NLP system. This limits both computation and storage.

- **Unknown Adaptation** A critical assumption of previous work is that a domain change has occurred. We must ascertain this ourselves.

Despite these challenges, we show unsupervised stream-based methods that effectively identify shifts in domain in language data. Furthermore, our methods are tied directly to the learning task so are sensitive to changes in actual task accuracy. Our methods have low false positive rates of change detection, which is important since examples within a single domain display a large amount of variance, which could be mistaken for a domain change.

Once a change is detected, any number of actions may be appropriate. The maintainer of the system may be notified that performance is suffering, labels can be obtained for a sample of instances from the stream for retraining, or large volumes of unlabeled instances can be used for instance reweighting (Jiang and Zhai, 2007).

3 Datasets

We begin the presentation of our methods by describing the data used in our experiments. We selected three data sets commonly used in domain adaptation: spam (Jiang and Zhai, 2007), ACE 2005 named entity recognition (Jiang and Zhai, 2007), and sentiment (Blitzer et al., 2007). Sentiment and spam are binary and ACE is multi-class. Note that in all experiments, a shift in the domain yields a decrease in system accuracy.

The goal of the spam data is to classify an email (bag-of-words) as either spam or ham (not-spam). Each email user may have different preferences and features. We used unigram and bigram features, following Dredze and Crammer (2008b) for feature extraction, and used the three task A users as three domains. The ACE 2005 named entity recognition dataset includes 7 named entity class labels (person, organization, location, geopolitical entity, facility, vehicle, weapon) for 5 text genres (newswire, broadcast news, broadcast conversations, conversational telephone speech, weblogs). We use 4000 examples from each genre and used Jiang and Zhai's feature-extracted data.[1] The sentiment data contains reviews from Amazon for four product types: books, dvds, electronics, and kitchen. We include an additional two types (music and video from Dredze and Crammer) in our false shift experiments and use unigram and bigram features, following Blitzer et al.

4 The $\mathcal{A}$-Distance

Our approach to detecting domain shifts in data streams that negatively impact system accuracy is based on the ability to (1) detect distributional changes in streams of real numbers and (2) convert document streams to streams of informative real numbers. This section describes how we achieve the former, and the next section describes the latter.

Theoretical work on domain adaptation showed that the $\mathcal{A}$-distance (Kifer et al., 2004), a stream based measure of difference between two arbitrary probability distributions P and P', can be used to evaluate the difference between two domain distributions (Ben-David et al., 2006). In a batch setting this corresponds to learning a linear classifier to discriminate the domains, and Blitzer et al. (2007) showed correlations with the error from domain adaptation. Given our interest in streaming data we return to the original stream formulation of $\mathcal{A}$-distance.

The $\mathcal{A}$-distance detects differences between two arbitrary probability distributions by dividing the range of a random variable into a set of (possibly

[1] We thank Jing Jiang for the feature-extracted ACE data.

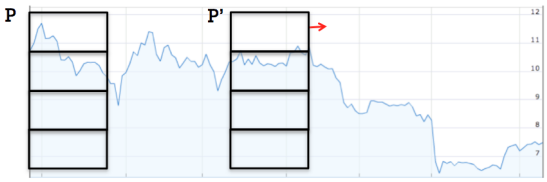

Figure 1: The $\mathcal{A}$-distance is computed between two windows (P and P' in a stream of real-valued data. The samples in each window are divided into intervals, and the $\mathcal{A}$-distance measures the change in the distributions over these intervals between the two windows.

overlapping) intervals, and then measures changes in the probability that a value drawn for that variable falls into any one of the intervals. If such a change is large, a change in the underlying distribution is declared. Let $\mathcal{A}$ be a set of real intervals and let $A \in \mathcal{A}$ be one such interval. For that interval, $P(A)$ is the probability that a value drawn from some unknown distribution falls in A. The $\mathcal{A}$-distance between P and P', i.e. the difference between two distributions over the intervals, is defined as follows:

$$d_{\mathcal{A}}(P, P') = 2 \sup_{A \in \mathcal{A}} |P(A) - P'(A)|.$$

Two distributions are said to be different when, for a user-specified threshold ϵ, $d_{\mathcal{A}}(P, P') > \epsilon$. The $\mathcal{A}$-distance is distribution independent. That is, it makes no assumptions about the form of the underlying distribution nor about the form of the change that might occur, either algorithmically or in the underlying theory. Unlike the L_1 norm, the $\mathcal{A}$-distance can be shown to require finitely many samples to detect distribution differences, a property that is crucial for streaming, sample-based approaches.

Since the $\mathcal{A}$-distance processes a stream of real numbers, we need to represent an example using a real number, such as the classification margin for that example. The first n of these numbers in the stream are a sample from P, and the most recent n are a sample from P'. We signal a domain shift when the $\mathcal{A}$-distance between P and P' is large (greater than ϵ). Larger values of n result in more accurate estimates of $P(A)$ and slower detection of changes.

The two windows of samples of size n are shown graphically in Fig. 1. Each increment on the horizontal axis represents the arrival of a new document. The vertical axis is some value computed from each document, such as its classification margin. To compute P and P', one needs to specify $\mathcal{A}$ and n, which are shown as two stacks of boxes that are identical except for their position. The width of each box is n, the number of examples used to estimate $P(A)$ and $P(A')$ for $A \in \mathcal{A}$, where the real interval A corresponds to the vertical span of the box. The value $P(A)$ is simply the number of documents whose real value falls inside that interval A divided by n. Note that the first n documents in the stream are used to compute P, and as each new document arrives the location of the stack of boxes used to compute P' is shifted to the right by one.

In Fig. 1, the number of examples whose real value falls in the top two intervals for P is approximately the same, with no example's value falling in the lower two intervals. For P', almost every one of the n document values falls in the second interval from the top, virtually assuring that $d_{\mathcal{A}}(P, P')$ will be large. Though the intervals in the figure do not overlap, they typically do.

Given n and intervals $\mathcal{A}$, the value of ϵ is chosen by randomization testing. Because the $\mathcal{A}$-distance is distribution independent, a sample of size $m \gg n$ is drawn from any distribution that spans $\mathcal{A}$. This sample is treated as a stream as described above, and the largest value of $d_{\mathcal{A}}(P, P')$ is stored. The sample is permuted and this process is repeated l times. Note that any change detection would be a false positive because all values were sampled from the same distribution. The values $d_{\mathcal{A}}(P, P')$ are sorted from largest to smallest, and ϵ is chosen to be the $\lfloor \alpha l \rfloor^{th}$ such value where parameter α is a user specified false positive probability.

Both the time and space complexity of our approach based on the $\mathcal{A}$-distance are small. Given n and $\mathcal{A}$, n instances must be stored in the sliding window and $2|\mathcal{A}|$ counters are required to represent P and P'. Note that both values are constants based on user specified parameters, not on the size of the stream. Processing a new instance involves computing its margin and updating P and P', all of which can occur in constant time.

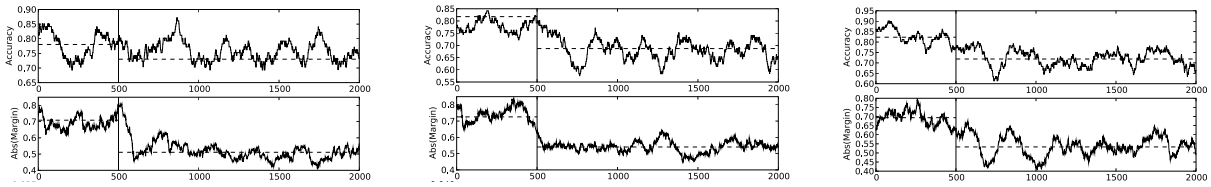

Figure 2: Each column of plots is a representative result using an SVM on a single run over a sentiment data shift: dvds → electronics, electronics → books, and kitchen → books, from left to right. The horizontal axis is the number of instances from the stream processed by the classifier. The top plot is the accuracy of the classifier on the last 100 instances. The bottom plot is the absolute value of the SVM classification margin. The vertical line at 500 instances marks the point of domain shift. Horizontal dotted lines indicate the mean of the accuracy/margin before and after the domain shift. Note that in all cases, the mean accuracy drops, as do the mean margin values, demonstrating that both can indicate domain shifts.

5 $\mathcal{A}$-Distance Over Margins

Since shifts in domains correlate with changes in distributions, it is natural to begin by considering the observed features in each example. When we shift from a source domain (e.g., book reviews) to a target domain (e.g., dvd reviews) we expect a change in the distribution for common source words ("author" and "plot" become less common). Since the $\mathcal{A}$-distance assumes a stream of single values, we can apply an $\mathcal{A}$-distance detector to each feature (e.g., unigram and bigram count) individually. However, our extensive experiments with this approach (omitted here) show that it suffers from a number of flaws, such as a high false positive rate if all features are tracked, the difficult problem of identifying an informative subset of features for tracking, and deciding how many such features need to change before a shift has occurred, which turns out to be highly variable between shifts.

Therefore, our goal is to use a single $\mathcal{A}$-distance tracker by collapsing each example to a single value. One way of doing this is to consider the *classification margin* produced by the classifier. The margin weighs features by their importance in classification. When more important features disappear, we expect the magnitude of the margin to decrease. Additionally, features that change but do not influence system performance are effectively ignored since they do not influence the margin. This approach has the advantage of task sensitivity, only tracking changes that impact task accuracy. Initial experiments showed effectiveness with the unsigned (absolute value of the) margin, which we use in all experiments.

We begin by examining visually the information content of the margin with regards to predicting a domain shift. The caption of Fig. 2 describes the setup, and the first row of the figure illustrates the effects of the shift on the source domain classifier's empirical accuracy, measured on a window of the previous 100 examples. The horizontal dashed lines indicate the average accuracy before and after the shift. Note that in each case, average classification accuracy drops after the shift. However, at any one point the accuracy displays considerable variance. Thus, while classification accuracy clearly suffers, it is difficult to measure this even in a supervised setting with labeled examples when considering a small portion of the stream.

The second row of Fig. 2 shows the average unsigned margin value of an SVM classifier computed over the previous 100 examples in the stream. The two dashed horizontal lines indicate the average margin value over source and target examples. There is a clear drop in the average margin value after the shift. This difference suggests that the margin can be examined directly to detect a domain shift. However, these values vary considerably so extracting useful information is not trivial.

We evaluated the ability of $\mathcal{A}$-distance trackers to detect such changes in margin values by simulating domain shifts using each domain pair in a task (books to dvds, weblogs to newswire, etc.). For each domain shift setting, we first trained a classifier on 1000 source domain instances. In our experiments, we used three different classification algorithms: Support Vector Machines (SVM) (Chang

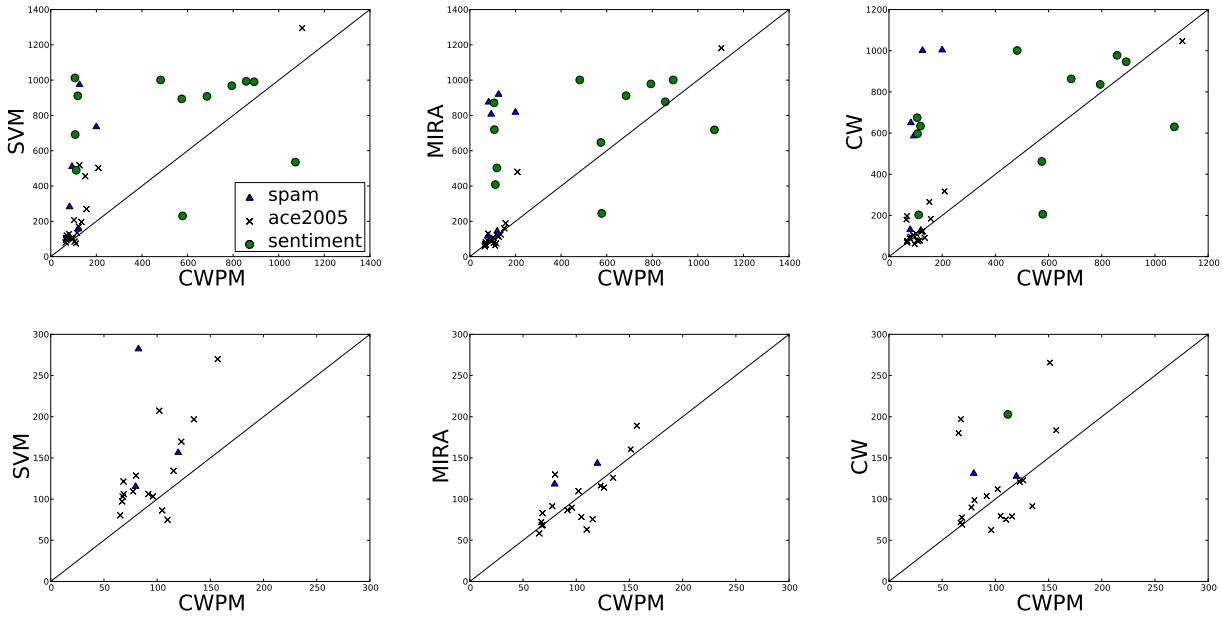

Figure 3: The mean number of instances after a domain change at which the $\mathcal{A}$-distance tracker detects a change. Each point represents the mean number of instances for CWPM (x-axis) and the SVM, MIRA and CW methods (y-axis). Datasets are indicated by different markers. The second row zooms each plot to the bottom left corner of the first row. Points above the diagonal indicate SVM, MIRA or CW took longer to detect a change than CWPM.

and Lin, 2001), MIRA (Crammer et al., 2006) and Confidence Weighted (CW) learning (Dredze et al., 2008). We evaluated each trained classifier on 500 test examples to measure accuracy on the source domain, and then used it to label examples in a stream. The first 500 examples in the stream were used for calibrating our change detection methods. The next 500 examples were from the source domain, followed by 1500 examples from the target domain. Over these 2000 examples we ran each of our detection methods. Experiments were repeated over 10 fixed random data permutations.

We automatically select $\mathcal{A}$-distance intervals as follows. First, we computed the mean and variance of the 500 calibration margins and then added intervals for .5 standard deviations away from the mean in each direction, .5 to 1 standard deviation in each direction, and intervals for 1 standard deviation to $\pm\infty$. We also added three evenly spaced overlapping intervals. To calibrate a FP rate of 0.05 we sampled from a Gaussian with the above mean and variance and used $n = 200$, $m = 10000$ and $l = 50$.

The results for each experiment (38 shifts repeated averaged over 10 runs each) are shown in Fig. 3.[2] Each plot represents one of the three classifiers (SVM, MIRA, CW) plotted on the vertical axis, where each point's y-value indicates the number of examples observed after a shift occurred before the $\mathcal{A}$-distance detector registered a change. Smaller values (lower points) are preferred. The second row of plots highlights the 0 to 300 region of the first row. (The x-axis will be discussed in the next section.) Notice that in many cases, a change was registered within 300 examples, showing that domain shifts can be reasonably detected using the margin values alone.

Equally important to detecting changes is robustness to false changes. We evaluated the margin detector for false positives in two ways. First, we logged any incorrectly detected changes before the shift. For all three algorithms, there were very few false positives (Table 1). The highest false positive rate was about 1% (CW), while for the SVM experiments, not a single detector fired prematurely in any experiment.

[2]The method plotted on the x-axis will be introduced in the next section. To evaluate the three methods in this section (SVM, MIRA, CW) compare the y-values.

Second, we sought to test the robustness of the method over a long stream of examples where no change occurred. In this experiment, we selected 11 domains that had a sufficient number of examples to consider a long stream of source domain examples.[3] Rather than use 500 source domain examples followed by 1500 target domain examples, all 2000 examples were from the source domain. All other settings were the same. For the SVM detector, out of 110 runs we detected 6 false positives, 3 of which were for the same data set (kitchen) (see Table 1.)

6 Confidence Weighted Margins

In the previous section, we showed that margin values could be used to detect domain shifts. We now explore ways to reduce the number of target domain examples needed to detect domain shift by improving the margin values.

Margin values are often taken as a measure of prediction confidence. From this perspective, the $\mathcal{A}$-distance margin tracker identifies when prediction confidence drops. Another task that relies on margins as measures of confidence is active learning, where uncertainty sampling for margin based systems is determined based on the magnitude of the predicted margin. Dredze and Crammer (2008a) showed how Confidence Weighted (CW) learning could be used to generate a more informative measure of confidence for active learning.

CW is an online algorithm inspired by the MIRA update (Crammer et al., 2006), which ensures a positive margin while minimizing parameter change. CW replaces the Euclidean distance used in the MIRA update with the KL divergence over Gaussian distributions. CW learning maintains a Gaussian distribution over linear weight vectors with mean $\boldsymbol{\mu} \in \mathbb{R}^N$ and diagonal covariance $\Sigma \in \mathbb{R}^{N \times N}$.

Maintaining a distribution over prediction functions is appropriate for our task where we consider margin values as confidence. We replace the margin $|\boldsymbol{w} \cdot \boldsymbol{x}|$, where $\boldsymbol{w}$ is a standard linear classifier, with a probabilistic margin $|\left(\Pr_{\boldsymbol{w} \sim \mathcal{N}(\boldsymbol{\mu}_i, \Sigma_i)}\left[\text{sign}(\boldsymbol{w} \cdot \boldsymbol{z}) = 1\right]\right) - \frac{1}{2}|$. Dredze and Crammer showed that this probabilistic margin can be translated into a *corrected* geometric margin, which is computed as the normalized margin as $\bar{\text{M}} = M/\sqrt{V}$, where M is the mean $M = \boldsymbol{\mu} \cdot \boldsymbol{x}$ and V the variance $V = \boldsymbol{x}^\top \Sigma \boldsymbol{x}$ of a univariate Gaussian distribution over the unsigned-margin $\text{M} = \boldsymbol{w} \cdot \boldsymbol{x}$. We call this method CWPM, for Confidence Weighted Probabilistic Margin.

We compared using CWPM to the standard margins produced by an SVM, MIRA and CW classifier in the last section. Fig. 3 shows the results of these comparisons. In each plot, CWPM (normalized margin) is plotted on the x-axis, indicating how many examples from the target domain were observed before the detector identified a change. The y-axis in each plot is the number of instances observed for the SVM, MIRA and CW methods. As before, each point is the average of the 10 randomized runs used above (assuming that detectors that did not fire do so at the end of the stream.) Points above the diagonal indicate that CWPM detected a change sooner than the comparative method. Of the 38 shifts, CWPM detected domain shifts faster than an SVM 34 times, MIRA 26 times and CW 27 times.

We repeated the experiments to detect false positives for each margin based method. Table 1 shows the false positives for the 38 domain shifts considered as well as the 11 false shift domain shifts. The false positive rates are among the lowest for CWPM. This shows that CWPM is a more useful indicator for detecting domain changes.

7 Gradual Shifts

We have shown detection of sudden shifts between the source and target domains. However, some shifts may happen gradually over time. We evaluate this by modifying the stream as follows: the first 500 instances come from the source domain, and the remaining 1500 are sampled randomly from the source and target domains. The probability of an instance being drawn from the target domain at time i is $p_i(\boldsymbol{x} = \text{target}) = \frac{i}{1500}$, where i counts from the start of the shift at index 500. The probability of sampling target domain data increases uniformly over the stream. At index 750 after the start of the shift each domain is equally likely. The ACE and Sentiment datasets had sufficient data to be evaluated in this setting. Fig. 4 shows CWPM still performs best, but results are close (SVM: 22 of 32, MIRA &

[3] ACE: bc, bn, cts, nw, wl; Sentiment: books, dvd, electronics, kitchen, music, video

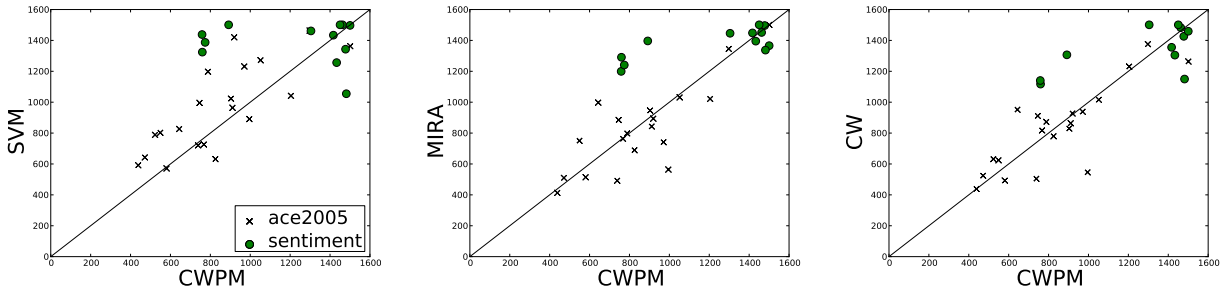

Figure 4: Gradual shift detection with SVM, MIRA or CW vs. CWPM. There were no false positives.

	Domain Shift FPs	
Algorithm	*True Shift*	*False Shift*
Sec. 5: SVM	0	6
Sec. 6: MIRA	2	13
Sec. 6: CW	5	10
Sec. 6: CWPM	1	6
Total tests	380	110

Table 1: False positives (FPs) observed in true domain shift and false domain shift experiments for methods in corresponding sections. Each setting was run 10 times, resulting in 380 true domain shifts and 110 false shifts.

CW: 17 of 32). As expected, detections happen later in the stream. The closer results are likely due to the increased difficulty of the task. With less clear information, there it is more difficult for all the algorithms to recognize a change, and performance across the methods begins to equalize. Even in this more difficult setting, CWPM is the best performer.

8 Comparison to Supervised Information

So far we have considered applying $\mathcal{A}$-distance tracking to information freely available in a real world system: the classification margins. As a useful baseline for comparison, we can measure using *supervised* sources of information, where additional information is provided that is not normally available. In particular, we investigate two types of supervised knowledge: the labels of examples in the stream and knowledge of the target domain. In each case, we compare using the $\mathcal{A}$-distance and CWPM versus applying the $\mathcal{A}$-distance to supervised information.

8.1 Classifier Accuracy

In Sec. 4 we showed that both the margin and recent classifier accuracy indicate when shifts in domains occur (Fig. 2). We developed techniques based on the margin, which is available at test time. We now consider knowledge of the true labels for these test examples, which allows for tracking classifier accuracy. We can use the $\mathcal{A}$-distance to detect when unexpected changes in accuracy occur.

For each test example classified by the system, we evaluated whether the system was correct in its prediction by examining the label. If the classifier was correct, we output a 1; otherwise, we output a 0. Over this 1/0 stream produced by checking classifier accuracy we ran an $\mathcal{A}$-distance detector, with intervals set for 1s and 0s (10,000 uniform samples to calibrate the threshold for a false positive rate of 0.05.) If an unusual number of 0s or 1s occur – more or less mistakes than on the source domain – a change is detected.[4] Results on this accuracy stream are compared to CWPM (Fig. 5.) Despite this supervised information, CWPM still detects domain changes faster than with labeled examples. Consider again Fig. 2, which shows both accuracy and margin values over time. While the average accuracy drops, the instantaneous value is very noisy, suggesting that even this additional information may not yield better domain shift detection. This will be interesting to explore in future work.

[4] An alternate approach would be to measure accuracy directly as a real valued number. However, our experiments showed the discrete approach to be more effective.

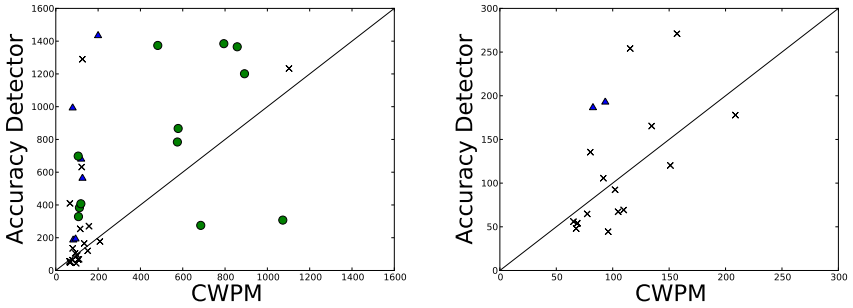

Figure 5: An $\mathcal{A}$-distance accuracy detector, run over a stream of 1s and 0s indicating correct and incorrect predictions of the classifier on examples in a stream. The bulk of points above the line indicate that CWPM is more effective at detecting domain change. CWPM had a single false positive and the accuracy detector had no false positives.

8.2 Domain Classification

Next, we consider another source of supervision: a selection of examples known to be from the target domain. In this setting, we know that a shift will occur and we know to which domain it will occur. This requires a sample of (unlabeled) target domain examples when the target domain is not known ahead of time. Using a common approach to detecting domain differences when data is available from both domains (Ben-David et al., 2009; Blitzer et al., 2007; Rai et al., 2010), we train a binary classifier to differentiate between the source and target domain. We learn a CW classifier on 1000 examples (500 from each domain) that do not appear in the test stream. We then label each example as either "source" or "target" and output a 1 or 0 accordingly. Over this 0/1 stream, we run an $\mathcal{A}$-distance detector with two intervals, one for 1s and one for 0s. The remaining setup is identical to the $\mathcal{A}$-distance experiments above.

Fig. 6 shows the detection rate of CWPM versus $\mathcal{A}$-distance over the domain classifier stream. As expected, the detection rate for the domain classifier is very fast, in almost every case (save 1) less than 400 examples after the shift happens. When CWPM is slow to detect a change (over 400 examples), the domain classifier is the clear winner. However, in the majority of experiments, especially for ACE and spam data, both detectors register a change quickly. These results suggest that while a sample of target domain examples is very helpful, our CWPM approach can also be effective when such samples are not available.

9 Related Work

Early NLP work in the *unsupervised* setting monitored classification confidence values, setting a confidence threshold based on a break-even heuristic, monitoring the rate of (presumed) irrelevant examples based on this threshold, and signaling a change when this rate increased (Lanquillon, 1999).

Confidence estimation has been used for specific NLP components such as information extraction. The correctness of fields extracted via a conditional random field extractor has been shown to correlate well to an estimate obtained by a constrained forward-backward technique (Culotta and McCallum, 2004). EM-based confidence estimation has been used to estimate the confidence of patterns derived from partially supervised relation extraction (Agichtein, 2006). Confidence estimation has also been used to improve the overall effectiveness of NLP systems. Confidence estimates obtained via neural networks have shown gains for speech recognition, spoken language understanding, and machine translation (Gandrabur et al., 2006). Pipeline models using confidence estimates at one stage as weights for further downstream stages improve over baseline dependency parsing and named entity recognition pipeline models (Bunescu, 2008).

An alternative formulation of domain adaptation trains on different corpora from many different domains, then uses linear combinations of models trained on the different corpora(McClosky et al., 2010).

Work in novelty detection is relevant to the task of detecting domain shifts (Scholkopf et al., 2000),

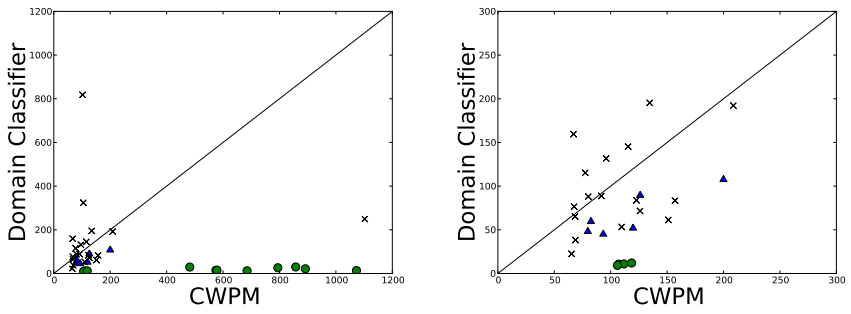

Figure 6: $\mathcal{A}$-distance over a stream of 1s and 0s produced by a supervised classifier trained to differentiate between the source and target domain. Samples from the unseen target domain is very effective. However, for many shifts, the margin based $\mathcal{A}$-distance detector is still competitive. CWPM had a single false positive while the domain classifier stream had 2 false positives in these experiments.

though the rate of occurrence of novel instances is more informative in our setting than the mere fact that novel instances are observed.

We are also motivated by the problem of detecting *genre shift* in addition to domain shift, as in the ACE 2005 data set shifts from newswire to transcripts and blogs. Different text genres occur in traditional settings, such as broadcast news transcripts and newswire, and have begun to proliferate with the variety of social media technologies now available including weblogs. Static genre classification has been explored using a variety of techniques, including exploiting punctuation (Kessler et al., 1997; Dewdney et al., 2001), TF-IDF statistics (Lee and Myaeng, 2002), and part-of-speech statistics and histograms (Finn and Kushmerick, 2006; Feldman et al., 2009).

Finally, statistical estimation in a streaming context has been considered in data mining applications (Muthukrishnan, 2005). Change detection via sequential hypothesis testing has been effective for streaming applications such as network intrusion detection (Muthukrishnan et al., 2007). Detecting new events in a stream of Twitter posts can be done using constant time and space similarity measures based on a modification of locality sensitive hashing (Petrović et al., 2010).

10 Conclusion

While there are a number of methods for domain adaptation, a system first needs to determine that a domain shift has occurred. We have presented methods for automatically detecting such domain shifts from a stream of (unlabeled) examples that require limited computation and memory by virtue of operating on fixed-size windows of data. Our methods were evaluated empirically on a variety of domain shifts using NLP data sets and are shown to be sensitive to shifts while maintaining a low rate of false positives. Additionally, we showed improved detection results using a probabilistic margin based on Confidence Weighted learning. Comparisons to detection with supervised information show that our results are effective even in unlabeled settings. Our methods are promising as tools to accompany the deployment of domain adaptation algorithms, so that a complete system can first identify when a domain shift has occurred before automatically adapting to the new domain.

Acknowledgments

Thanks to the HLTCOE text processing group for many helpful discussions.

References

Eugene Agichtein. 2006. Confidence estimation methods for partially supervised information extraction. In *SDM*.

Shai Ben-David, John Blitzer, Koby Crammer, and Fernando Pereira. 2006. Analysis of representations for domain adaptation. In *NIPS*.

Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Vaughan. 2009. A theory of learning from different domains. *Machine Learning*.

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *EMNLP*.

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, Bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Association for Computational Linguistics (ACL)*.

Razvan C. Bunescu. 2008. Learning with probabilistic features for improved pipeline models. In *EMNLP*.

Chih-Chung Chang and Chih-Jen Lin, 2001. *LIBSVM: a library for support vector machines*. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online passive-aggressive algorithms. *Journal of Machine Learning Research (JMLR)*.

Aron Culotta and Andrew McCallum. 2004. Confidence estimation for information extraction. In *North American Chapter of the Association for Computational Linguistics - Human Language Technologies (NAACL-HLT)*.

Hal Daumé. 2007. Frustratingly easy domain adaptation. In *Association for Computational Linguistics (ACL)*.

Nigel Dewdney, Carol VanEss-Dykema, and Richard MacMillan. 2001. The form is the substance: classification of genres in text. In *Workshop on Human Language Technology and Knowledge Management*.

Mark Dredze and Koby Crammer. 2008a. Active learning with confidence. In *Association for Computational Linguistics (ACL)*.

Mark Dredze and Koby Crammer. 2008b. Online methods for multi-domain learning and adaptation. In *EMNLP*.

Mark Dredze, Koby Crammer, and Fernando Pereira. 2008. Confidence-weighted linear classification. In *ICML*.

S. Feldman, M. A. Marin, M. Ostendorf, and M. R. Gupta. 2009. Part-of-speech histograms for genre classification of text. In *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*.

Jenny Rose Finkel and Christopher D. Manning. 2009. Hierarchical Bayesian domain adaptation. In *NAACL-HLT*.

Aidan Finn and Nicholas Kushmerick. 2006. Learning to classify documents according to genre: Special topic section on computational analysis of style. *J. Am. Soc. Inf. Sci. Technol.*, 57(11):1506–1518.

Simona Gandrabur, George Foster, and Guy Lapalme. 2006. Confidence estimation for NLP applications. *ACM Trans. Speech Lang. Process.*, 3(3):1–29.

Jing Jiang and ChengXiang Zhai. 2007. Instance weighting for domain adaptation in NLP. In *Association for Computational Linguistics (ACL)*.

Brett Kessler, Geoffrey Numberg, and Hinrich Schütze. 1997. Automatic detection of text genre. In *Association for Computational Linguistics (ACL)*.

Daniel Kifer, Shai Ben-David, and Johannes Gehrke. 2004. Detecting change in data streams. In *Very Large Data Bases (VLDB)*.

Ralf Klinkenberg and Thorsten Joachims. 2000. Detecting concept drift with support vector machines. In *International Conference on Machine Learning (ICML)*.

C. Lanquillon. 1999. Information filtering in changing domains. In *IJCAI*.

Yong-Bae Lee and Sung Hyon Myaeng. 2002. Text genre classification with genre-revealing and subject-revealing features. In *SIGIR*.

David McClosky, Eugene Charniak, and Mark Johnson. 2010. Automatic domain adaptation for parsing. In *NAACL-HLT*, pages 28–36, Los Angeles, California, June. Association for Computational Linguistics.

S. Muthukrishnan, Eric van den Berg, and Yihua Wu. 2007. Sequential change detection on data streams. In *IEEE International Conference on Data Mining Workshops (ICDMW)*.

S. Muthukrishnan. 2005. Data streams: Algorithms and applications. *Foundations and Trends in Theoretical Computer Science*, 1(2).

Kyosuke Nishida and Koichiro Yamauchi. 2007. Detecting concept drift using statistical testing. In *Discovery Science*.

Kyosuke Nishida. 2008. *Learning and Detecting Concept Drift*. Ph.D. thesis, Hokkaido University, Japan.

Saša Petrović, Miles Osborne, and Victor Lavrenko. 2010. Streaming first story detection with application to twitter. In *NAACL-HLT*, pages 181–189, June.

P. Rai, A. Saha, H. Daumé III, and S. Venkatasubramanian. 2010. Domain Adaptation meets Active Learning. In *Workshop on Active Learning for Natural Language Processing (ALNLP)*, page 27.

Bernhard Scholkopf, Robert Williamson, Alex Smola, John Shawe-Taylor, and John Platt. 2000. Support vector method for novelty detection. In *NIPS*.

Gerhard Widmer and Miroslav Kubat. 1996. Learning in the presence of concept drift and hidden contexts. *Machine Learning*, 23:69–101.