Blog Classification with Co-training

Joshua Gerrish, Vahed Qazvinian, Xiadong Shi

December 15, 2007

Abstract

In this project we use co-training to classify blogs by political leaning. We classify them into two pre-defined categories: liberal and conservative. We examine the performance of co-training versus normal supervised learning. We also look at using several different features to improve training.

1 Introduction

The aim of this project is to use co-training to classify the political leaning of blogs. Co-training was first introduced by Blum and Mitchell as a method of exploiting distinct views of a dataset to help classify a large set of unlabled data from an initial small set of labeled data [4]. To demonstrate co-training, they created a binary classification problem for webpages, where the target class is "course homepage". A webpage can be either a course homepage or some other type of page.

The web provides a huge collection of data for machine learning algorithms. Current estimates place the number of web pages at 11.5 billion [8]. Unfortunately, most of this data is unlabled. Using an algorithm like co-training, we can make use of a small number of labeled samples to classify a much larger amount of information.

Recently, the web has seen an explosion of weblogs, or blogs. These are personal diaries that people maintain online. Like a traditional diary, people can make multiple entries, which are commonly called posts or blog posts. The content of these weblogs usually consists of personal entries stating an individual's opinion about a subject. Because of the personal nature of weblogs, they provide useful data about social systems. A person who maintains a blog is called a blogger.

Many people post their feelings about current political happenings. This was especially apparent during the 2004 United States Presidential Election. Adamic and Glance [2] looked at political blogs, blogs that are dedicated primarily to political dealings, during this time period. They found a natural split between liberal and conservative bloggers. This split manifested itself in multiple ways, although their research primarily dealed with the network structure of blogs.

Hyperlinks connect the web, any webpage can link to any other webpage with a hyperlink. Hyperlinks appear in browsers are snippets of text that when clicked on lead the user to another webpage. The web can thus be viewed as a large directed graph, where webpages are nodes and the links are edges.

We are trying to use the co-training technique to classify blogs by their political leaning, either liberal or conservative, which can be considered a binary classification problem. Since labeling blogs is a labor-intensive process, we can benefit from the semi-supervised nature of co-training. In this project we try to investigate whether co-training will improve the blog classification problem.

Our innovation in this project is two-fold: Firstly, we apply co-training in a new context, classifying political leaning. Secondly, we evaulate co-training against several different methods of web page classification.

1.1 Motivation

We aim to test the use of co-training on blog datasets by comparing the results of single classifiers based on the content of the blogs or the link structure with those of the co-trainer.

In a co-training classifier, there are two simple classifiers (usually Naïve Bayes) which are initially trained with a small set of labeled data. Each classifier is trained on a different subset of the training data's features. The co-training algorithm is supposed to gradually extend the training data with self-labeled data.

1.2 Related Work

Traditional machine learning techniques often require a high volume of labeled training data to be instantly available for the training algorithm. However in a real learning context, this is often not the case. For example, in tasks of classifying Web pages, the collection of documents that we need to classify is vast, while the number of available labeled training data is extremely limited. Another example is the segmentation of Web search engine query logs, in which acquiring a training sample set that is large enough is difficult if not impossible.

Co-training algorithms [4] were developed for the purposes of boosting the performance of learning algorithms given insufficient training data. In a two-classifier setting, the view of the data set, i.e. the set of all available features, is split into two subparts, each rendering a input feature set to one of the classifiers. Normally, the two sets of views are semantically different. For example, the two different views could be the content of web pages and the text of links pointing to them. Another example is in image classification, with the two views being pixel data and picture caption.

Each classifier is trained on one feature set only. Initially, only a small number of labeled training samples are available. When a classifier is trained, it is used to predict the labels of the unlabeled data and assign confidences to their predictions. The top few samples that are predicted to be positive or negative with high-confidences are then selected and removed the unlabeled data set. They are merged into the labeled training data set and together used for training the other classifier. Recursively, the smaller set of labeled training samples is augmented by a much larger and inexpensive unlabeled samples at a low cost. Blum and Mitchell [4] reported a significant performance improvement on classifying Web pages by adopting co-training on Naïve Bayes classifiers. Kiritchenko and Matwin [10] implemented co-training methods on top of Naïve Bayes classifiers as well as Support Vector Machine (SVM) classifiers and reported increased performance for both classifiers.

Co-training is somewhat similar to Yarowsky's approach to the word sense disambiguation task [14]. Some words have multiple meanings, some quite different from another. An example is "banK", which can be a verb, as in "the plane banked to the right", or a noun, as in "I deposited my money in the bank." The word sense disambiguation problem is to determine the sense of a word given it's surrounding context or use. Yarowsky observed that in a single document, all instances of a word are likely to have the same meaning. This is like co-training, where one view is the document, and the other view is the word's context.

1.3 Machine learning and the web

There has been quite a bit of work using the web as a source of data for learning tasks. One of the ultimate goals is described by Tim Berners-Lee, creator of the World Wide Web as the Semantic Web [3]. The web pages in the semantic web will be readable by both humans and machines. Software agents will be able to automatically combine data from different sources and reason about this data. Using machine learning, we can automatically generate machine readable content from the existing web, bootstrapping the semantic web [5].

There has been some research looking specfically at classifying blogs. One area of research is looking at spam detection in blogs. Since the cost of starting a blog is so low, it is relatively easy to setup a spam blog or splog. Kolari et al used SVMs to detect spam blogs with accuracy around 88%.

There has also been research into classifying blogs by the topic of the blog. Qu et al classified into a predefined set of four categories: personal, news, political or sports. They used a Naïve Bayes classifier trained on the first sentence in the content, title, and anchor text [12].

The problem we are looking at, classifying blogs according to political leaning or cultural orientation has also been looked at before. Heckel and Ward used self-organizing maps to classify blogs according to political orientation with very good success [9]. Malouf and Mullen classified users according to political affiliation using text methods and by generating a graph based on the quoting patterns of users. If a user quotes another user, a link is made between the first user and the second user [11]. Efron also used Naïve Bayes, SVMs and link information to classify web documents and blogs by political orientation [6]. His work presents some new algorithms for using the link structure to classify web pages.

2 Data

We are using the data set from Adamic and Glance [2]. Our training set includes a sample of 1494 political blogs extracted from the Web, labeled as either ”conservative” or ”liberal.” We have the posting history for a two week period from 40 of these blogs: 20 conservative and 20 liberal. We also have access to the an archive of blog posts going back several years for the larger data set. We can extract additional posts from the larger set of 1494 labeled blogs if needed. This data is available online [1].

2.1 Structure

Our data set is based on blog posts. A single blog can have multiple blog posts. Each blog post can have several links.

- **Content**
  The blog content is the first feature space we decided to choose. The content is text, which we treat as a bag-of-words, ignoring word order. We generate vectors representing each blog, where each entry corresponds to a unique word. The size of all vectors is the same for all blogs. The entries in the vector are tf-idf values [13].
- **Anchor Text**
  Each link on a webpage can have text associated with it. This usually shows up as underlined text of a different color. This text is called the anchor text. Previous research has shown the importance of citaiton summaries. In fact if blog A is pointing to B in a sentence that sentence might be relevant to the pointed blog. Thus a set o fsuch sentences can be a good feature to analyze. As the content, the anchor texts are in pure text format and can similary be re-presented and indexed.

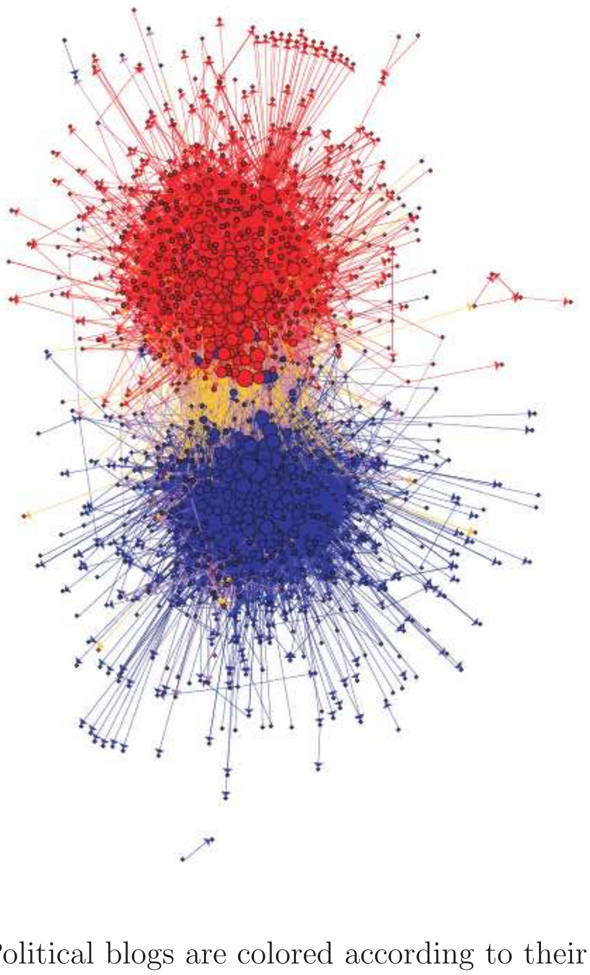

Figure 1: Political blogs are colored according to their leanings

- **Outgoing Link**
  Link structure of the blog has already been studied in previous research [7]. It is likely that similar blogs tend to link to each other, and thus the link structure can be used to classify blogs' leaning. An *OutGoing* link points from a blog to another blog. We want to investigate how important link structure is in classifying blogs.
- **Incoming Link**
  Incoming link is the set of blogs that link to a specific blog. We want to investigate if the leaning of a blog can be predicted based on who are linking to it. That it, If a greater number of liberal blogs are pointing to a blog, will that mean the target blog is liberal or not.

2.2 Preprocessing

The blog data is in XML format and includes information such as the date the post was crawled, post title, author, URL, entry content and the blog name. The blog posts contain HTML formatting, which we removed.

The sanitized posts of each blog are then concatenated together into a single large document containing all posts for that blog. The text is split into words, a process called tokenization. These words can be combined into a single term vector. The entries in the vector correspond to the number of occurrences of that term in all posts for a specific blog. A term vector is generated for each blog we found posts for. If a blog did not contain a term, the corresponding entry in the vector is zero.

The anchor text representation of a blog is generated by aggregating all the anchor texts of hyperlinks that point to a certain blog. Other textual attributes of blogs are processed in similar ways. The hyperlink structure of the blogsphere is constructed rather differently, by connecting two blogs if and only if there is one or more hyperlink (identified from the blog's textual contents) pointing from one of them to the other. The weights of the arcs are associated with the number of hyperlinks pointing to each other.

The entire set of 1494 blog posts is divided into "conservative" and "liberal" classes. We assign positive labels (+1) to "conservative" and negative labels (-1) to "liberal". The leanings of the blogs are already annotated manually by human evaluators prior to our work, and are marked up in the XML data.

There are a total of 59 blogs and 50136 unique words in the cleaned data set.

3 Method

We characterize blogs in terms of different views, such as the blog text, incoming/outgoing link structure and anchor texts. Due to the different natures of these views, there is no previous work to build a single classifier that can capture and combine all of these views and work based on top of the combined feature sets in an interpretable way. For each view, we can convert it into a feature set, and represent blogs in terms of such a feature set, base the selected classifier on it for training purposes. The training dataset consists of two parts, with the first part being labeled samples (the number of which is usually small) while the second being unlabeled samples. All classifiers are trained initially on the given labeled samples, based on their corresponding feature set (i.e. view of the data). Then each classifier is run on the unlabeled training examples, assigning labels to them. A subset of these examples, which are predicted to be positive (or negative) with high degrees of confidence by the classifier, are selected as the co-training examples, removed from the unlabeled training data set and mixed with the existing labeled training examples. The updated labeled examples are then feed as the input to the next classifier, which again produces labeling of the remaining unlabeled examples, selects high-confidence predictions and updates the available labeled samples. Repeatedly, high-confidence prediction results of one classifier are plugged (together with existing labeled data) into the other classifier(s), which provides a much larger base of training data than the original labeled data.

A very important assumption of co-training is the conditional independence assumption. If X is our input space, and X_1 and X_2 are partitions or views of that data set Y is our label, conditional independence implies that $Pr[X_1 = x_1|X_2 = x_2, Y = y] = Pr[X_1 = x_1|Y = y]$. That is, given $Y = y$, the X_1 and X_2 are conditionaly independent. In our case, it is intuitive that the probability distributions of terms in textual contents of blogs are conditionally independent of the anchor texts of these blogs, because different agents create the incoming link and content.

Input::

- $F = (F_1, F_2, ., F_m)$ which are m semantically different feature sets (that can be considered as different views of the data)
- $C = (C_1, C_2, ., C_m)$ which are m supervised learning classifiers (each of which corresponds to a unique feature set)
- L is a set of labeled training data
- U is a set of unlabeled training data

Procedure of Co-training::

- Training each classifier C_i initially on L with respect to F_i
- For each classifier C_i:
  - C_i labels the unlabeled training data from U based on F_i
  - C_i chooses the top p positive and top n negative labeled examples E from U according to the confidence of the prediction
  - Remove E from U
  - Add E into L with corresponding labels predicted by C_i

We set $p = 1$ and $n = 1$ for our experiments. In each step, each classifier removes at most a single unlabled data point and inserts it into the labeled data L. The algorithm terminates when there has been no change in L and U.

The actual co-training algorithm that we adopted is a modified version based on the work by Kiritchenko and Matwin [10]. There are several elements that make our method novel. First, we render more than two feature vectors for co-training, each corresponding to a single classifier. The original work by Blum and Mitchell used out-going link text (anchor text) and the content of the page. We add more features into the feature base, including the hyperlink structure of the blogsphere and the titles of the blogs. Second, in co-training we implement different types of classifiers that are naturally suitable for different feature sets. For example, kNN reasonably fits into the hyperlink structure of blogsphere , because in the hyperlink structure the distances between blogs can be naturally defined as the number of hyperlinks pointing to each other, while the representations of blogs themselves are not necessarily important or needed. Previous work have little discussion on how different classifiers can be incorporated in co-training.

3.1 Basic classifiers

Naïve Bayes is often used as the baseline classifier in text classification because it is fast and easy to implement. In this project we decided to use the Naïve Bayes classifier as the underlying text classifier.

The basic Naïve Bayes classifier works by finding the most likely class for a document. The probability of a document d being in a class c is given by equation 1:

$$P(c|d) = P(c) \prod_{1 \leqslant k \leqslant n_d} P(t_k|c) \tag{1}$$

$P(t_k|c)$ is the probability that term k appears in documents of class c. For classification, we want to find the most likely class, so we find the maximum probability over all classes. Because of the possibility of floating point errors, we use the log likelihood, as given in equation 2.

$$class = \arg\max_{c \in C} [\log \hat{P}(c) + \sum_{1 \leqslant k \leqslant n_d} log \hat{P}(t_k|c)] \tag{2}$$

In this case, $\hat{P}(c)$ is the relative frequency of documents of class c in the entire corpus. If there are N documents and N_c documents of class c in the entire corpus, $\hat{P}(c)$ is:

$$\hat{P}(c) = \frac{N_c}{N} \tag{3}$$

$\hat{P}(t_k|c)$ is the relative frequency of term t in all documents of class c, as shown in equation 4. V is the vocabulary size, or th enumber of unique terms in the corpus.

$$\hat{P}(t_k|c) = \frac{T_{ct}}{\sum_{t\prime \in V} T_{ct\prime}} \tag{4}$$

One problem with this estimate is that terms that occur in some classes but not others will get a probability of zero because of data sparseness. We can alleviate this issue using add-one smoothing. We add one to every term count and normalize, as shown in equation 5.

$$\hat{P}(t_k|c) = \frac{T_{ct} + 1}{\sum_{t\prime \in V} T_{ct\prime} + |V|} \tag{5}$$

We also use the k-Nearest Neighbor classifer to classify blogs given the link structure. A blog is classified based on the majority-vote of your neighbors. For example, if four conservative and three liberal blogs link to you, the blog is labeled conservative.

4 Experiments

For our experiments, we followed the proceess in [4]. We first selected 14 of the blogs as training data. This corresponds to approximately 25% of the total data. The remaining data was used to generate a labeled set L and an unlabeled set U.

As an evaluation framework, we split the entire dataset into two portions, one being the labeled training samples **S** while the other being unlabeled test data set **T**. The labeled training samples are further divided into two parts, with a first part S_1 representing the initial training set while the second part S_2 being the co-training sample set. We first train the classifiers separately on S_1 and test them individually on T. We then implement the co-training algorithm on top of these classifiers as described previously. We then initialize the co-training algorithm on S1 and let it train itself on S_2. The resulting classifiers from co-training are then tested on T. We conduct a number of experiments with different compositions of S_1, S_2 and T, in order to get a comprehensive performance evaluation of the classifiers. The performance of each classifier, when run separately, is compared with the performance of the same classifier resulting from co-training, with the same experimental setting.

Classifiers	Feature Space	Test Size	$\|L\|$	$\|U\|$	Accuracy
Naïve Bayes	Anchor Text	14	38	n/a	59.52%
Naïve Bayes	Content	14	38	n/a	80.61%
KNN	Outgoing links	14	38	n/a	79.59%
KNN	Incomming links	14	38	n/a	81.63%
Co-training (Naïve Bayes)	Anchor text + Content	14	38	7	78.57%
Co-training (KNN)	Outgoing link + Incoming link	14	38	7	85.71%
Co-training (Naïve Bayes, KNN)	Anchor text + Incoming link	14	38	7	64.28%
Co-training (Naïve Bayes, KNN)	Anchor text + Outgoing link	14	38	7	71.43%
Co-training (Naïve Bayes, KNN)	Content + Incoming link	14	38	7	80.00%
Co-training (Naïve Bayes, KNN)	Content + Outgoing link	14	38	7	76.67%

Table 1: Comparison of the performance of Naïve Bayes Classifier with/without co-training.

The performance of classifiers is measured in terms of prediction error rate. The prediction error rate is calculated as the probability that a given blog is misclassified. This is equal to the number of misclassified blogs divided by the total number of blogs classified by the classifier.

5 Results

Results are given in table 1. In our test data there are 59 blogs: 34 conservative(+1), and 25 liberal(−1).

Figure 2 shows the performance of three different classifiers. The solid and dotted lines show the performance of KNN on the incoming and outgoing link data respectively. The dashed green line shows the performance of the co-training classifier. One thing to note is that KNN performs very poorly when there are very few training points. Our implementation does not label a blog when it has no labeled neighbors. In the co-training code, all blogs are labeled. For very small training size runs, this results in close to 50% accuracy.

6 Conclusion

Due to our small sample size, the performance of co-training was not as high as we would have liked. We are currently generating a new sample set that contains many more blogs. We are also looking at using indivdual blog posts as data points instead of combining all posts for a blog into a single data point.

Even with the small data set, we did see a good performance with co-training using link structure and content. We found the anchor text was a poor indicator of political orientation.

7 Future Work

We would like to explore using other classifiers such as SVM. We would also like to compare the dataset we have from 2004 with current blog data, to see how classifiers trained with older content perform with newer blog posts.

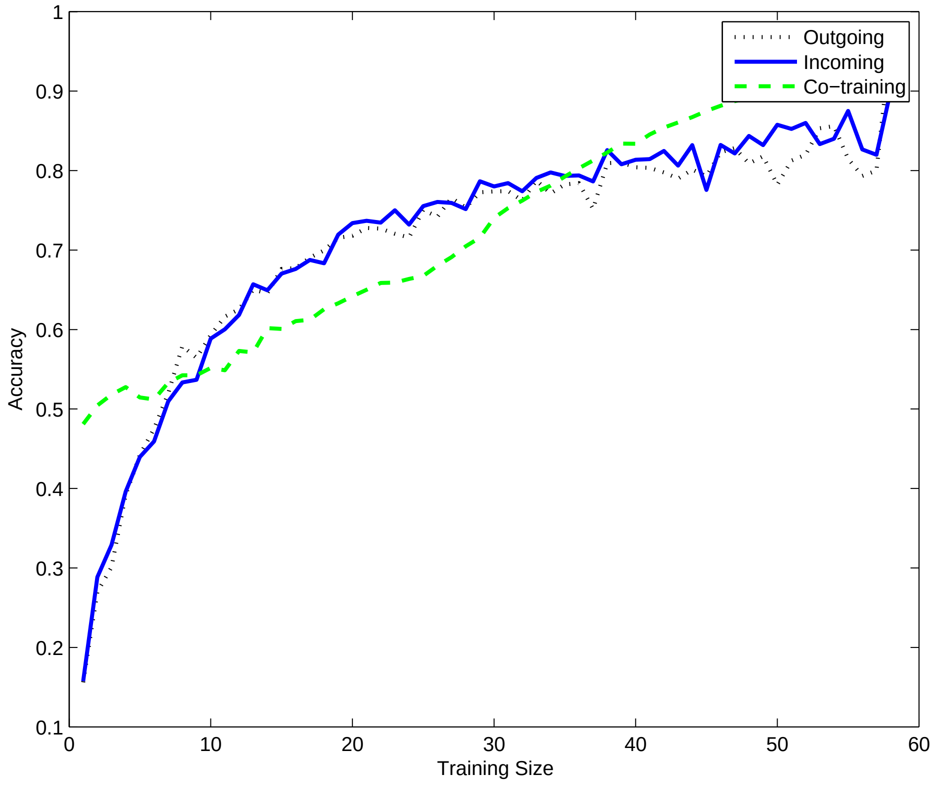

Figure 2: Performance of KNN and co-training on link structure data.

8 Individual Contribution

- **Joshua Gerrish**
  Josh wrote most of the parsing codee and preprocessing code. He also wrote the Naïve Bayes code and some experimental framework. For the paper he wrote the related work section and preprocessing.
- **Vahed Qazvinian**
  Vahed wrote the co-training code and the kNN implementation in MATLAB. He also wrote most of the experimental framework. For the paper, Vahed wrote most of the initial paper and
- **Xiaodong Shi**
  Xiaodong helped with the mid-project report and helped with the preprocessing.

References

[1] Mark newman's network data.

[2] Lada Adamic and Natalie Glance. The political Blogosphere and the 2004 U.S. election: Divided they Blog. In *Proceedings of the WWW '05 Second Annual Workshop on the Weblogging Ecosystem: Aggregation, Analysis, and Dynamics*, 2005.

[3] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, 284:28–37, 2001.

[4] Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the 11th Annual Conference of Learning theory*, pages 92–100, July 1998.

[5] Mark Craven, Dan DiPasquo, Dayne Freitag, Andrew McCallum, Tom Mitchell, Kamal Nigam, and Sean Slattery. Learning to extract symbolic knowledge from the world wide web. pages 509–516, Madison, Wisconsin, United States, 1998. American Association for Artificial Intelligence.

[6] M. Efron. Cultural orientation: Classifying subjective documents by cociation analysis. *AAAI Fall Symposium on Style and Meaning in Language, Art, and Music*, 2004.

[7] Eric J. Glover, Kostas Tsioutsiouliklis, Steve Lawrence, David M. Pennock, and Gary W. Flake. Using web structure for classifying and describing web pages. pages 562–569, Honolulu, Hawaii, USA, 2002. ACM.

[8] A. Gulli and A. Signorini. The indexable web is more than 11.5 billion pages. pages 902–903, Chiba, Japan, 2005. ACM.

[9] F. Heckel and N. Ward. Political blog analysis using bootstrapping techniques. *Class of 2005 Senior Conference on Natural Language Processing.*

[10] Svetlana Kiritchenko and Stan Matwin. Email classification with co-training. In *CASCON '01: Proceedings of the 2001 conference of the Centre for Advanced Studies on Collaborative research*, page 8. IBM Press, 2001.

[11] R. Malouf and T. Mullen. Graph-based user classification for informal online political discourse.

[12] H. Qu, A. La Pietra, and S. Poon. Classifying blogs using nlp: Challenges and pitfalls.

[13] Gerard Salton and Michael J. McGill. *Introduction to Modern Information Retrieval.* McGraw-Hill, Inc., New York, NY, USA, 1986.

[14] David Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. pages 189–196, Cambridge, Massachusetts, 1995. Association for Computational Linguistics.