Using Support Vector Machines to Identify Personal Blog Entries

Khai Tran
Department of Computer Sciences, UW-Madison

Abstract

One of the problems in navigating the content of a blog is the interference of personal entries among informative entries, which makes the readers waste a considerable amount of time to identify and skip these entries. In this project, we leverage the Support Vector Machines technique to identify personal entries from a blog.

Introduction

Weblog, or blog, is a type of websites maintained by an individual with regular entries of commentary (**?**). Blogs are becoming an important source of data on the Internet as they are providing various kinds of information, ranging from commentary or news on a particular subject to more personal information like online diaries. Popular sites for hosting a blog are Myspace, Blogspot, Wordpress, Yahoo! blog,

However, the problem in navigating the content of a blog is that it often contains a lot of personal information about bloggers, such as what they have done in the day, how much they love their puppies, or so on. As a reader, we probably do not want to waste our precious time for reading such kinds of information. Thus, it would be useful if we have a tool for automatically detecting personal entries and get rid of them.

In this project, we leverage the Support Vector Machines (**?**) technique identify personal entries from a blog.

Approach

Among different classification techniques, we choose Support Vector Machines because it's easy to use and can bring a high accuracy. In addition, SVM-light software package (**?**) , which we use in our experiments, is a well-known package and used widely in the research community.

We treat each blog entry as a bag of words, and represent it as a vector in the feature space with each word type as a dimension. Then we use a dataset, described in the next section, to train the model.

After getting the SVM parameters, we can integrate the model into a RSS feeder. Each time the feeder crawl a blog, it uses the model to detect where an entry has personal content or not. If yes, it will treat that entry as a "spam" one and ignore it.

Dataset

Data sources

To train the SVM model, we need two sets of data, the positive set with personal entries and the negative set with non-personal entries. If we use common blog sites where a blog may contain both personal and non-personal entries, such as Blogspot or Myspace, we have to manually go over each entry to read and decide whether it is personal or not. Using this way to obtain a large dataset is a very time-taking job. Instead, we select "special" blog sites where they contain only either personal or non-personal entries.

For positive dataset, we download 200 random entries from the Open Diary (**?**). Entries from Open Diary mostly contain personal information of individuals, including what they are thinking of, what they have done in the day...

For negative dataset, we select 200 entries from the New York Times blog (**?**). To guarantee the diversity of information, these entries are picked up from 20 different topics, each topic with 10 entries. These topics include arts, cultures, world, local, magazine, technology, science, food, movie, education, magazine, baseball, old age, business, economics, traveling, environment, college sport, design, and animation.

Text preprocessing

Text from each blog entry needs to be preprocessed before feeding into the SVM training program. First, we run sentence segmentation program to detect the sentence boundaries. Then, we use a word tokenizer to separate individual words. Finally, we use stemming technique convert all words to the standard lower case letter forms. After the preprocessing step, we obtain two data files, one for positive sample and the other for negative samples. In each file, each line represents a document (a blog entry). From these files, we can easily convert them to the svm-light formats used for training the SVM model.

Data Summary

Table 1 shows the summary of the our datasets:

Parameter	Value
Number of negative docs	200
Number of positive docs	200
Vocabulary size	10,356
Total no. of word tokens	131,562
Ave. size of negative docs	270
Ave. size of positive docs	388

Table 1: Dataset summary

Experiments and results

Experiment settings

We do two experiments. In the first experiment, we exam the accuracy of the model with the given dataset using 10-fold cross validation. The dataset is divided randomly into 10 folds, each fold with 20 negative and 20 positive documents. Then, we sequentially use one fold for testing and the nine remaining folds for training.

We run the experiment four times. For the first three runs, we use linear kernel but with different random folds to obtain the accuracies as well as stability of the results. In the the fourth run, we keep the random fold from the third time and use the polynomial kernel with $c = 1, s = 1, d = 2$.

In the second experiment, we try to predict the representative words from each set of documents. We use the whole dataset to train the model, and from the parameter vector w obtained after the training process, we look up words that have the corresponding smallest (negative) and largest (positive) values in the vector. Words with smallest (largest) are the representative words for negative (positive) documents.

Experiment 1: Model accuracy

Results of the first experiments is shown in Table 2

testing fold	1st run	2nd run	3rd run	4th run
1	95	92.5	92.5	85
2	87.5	90	92.5	80
3	87.5	87.5	82.5	65
4	87.5	82.5	90	77.5
5	87.5	90	85	80
6	82.5	97.5	92.5	85
7	90	90	85	75
8	87.5	90	80	70
9	87.5	80	92.5	85
10	85	85	92.5	82.5
average	87.75	88.5	88.5	78.5

Table 2: The model accuracy

The experiment results show that for the given dataset, the accuracy of the model are fiarly high and stable, and the linear model is more accurate than the polynomial kernel.

Experiment 2: Representative words

Table 3 and 4 show top words appearing in positive and negative documents, respectively. We can see that except top-10 words in both documents, which are actually stop words, representative words from negative documents are somewhat more formal than representative words from positive documents. This result is understandable since positive documents are entries about personal diary, while negative documents are entries about news or commentaries on New York Times.

Rank	Words
1-10	'i' "s' '...' 'it' 'n't' 'you' '.' '!' 'to' 'we'
11-20	'so' 'me' "" 'have' 'all' "" "m' "" 'will' 'do'
21-30	'realli' 'go' 'need' 'then' 'out' 'he' 'your' 'eat' 'him' 'just'
31-40	'than' 'leav' 'try' 'todai' 'healthi' 'friend' 'up' 'note' 'am' 'like'
41-40	'around' 'get' 'feel' 'god' 'be' 'person' 'when' 'back' '=' 'never' 'night'

Table 3: Representative words in positive documents

Rank	Words
1-10	',' 'of' 'the' 'in' '' 'a' '' '(' 'from' 'with'
11-20	')' 'its' 'by' 'film' 'my' 'as' 'citi' 'their' 'post' 'at'
21-30	'realli' 'about' 'list' 'comment' 'number' 'mr.' 'call' 'and' 'ad' 'percent' 'us'
31-40	'wont' 'here' 'price' 'creat' 'public' 'week' 'two' 'wa' 'son' 'doesnt'
41-50	'financi' 'new' 'column' 'iphon' 'ramp' 'dont' 'report' 'blog' 'design' 'york'

Table 4: Representative words in negative documents

Related work

There are several studies on blog identification and classification. In (**?**),Kolari et al. use Support Vector Machines to identify a blog in the blogsphere. In other direction, Sun et al. leverage the blog tags to increase the accuracy in blog topic classification (**?**).

Conclusions and future work

The results from our model demonstrate that the Support Vector Machines model work well in the personal blog identification problem, with fairly high accuracy and understandable results. However, we still have several limitations in our method.

First, we have not removed the stop words from the dataset. That is the reason why stop words appear in the top lists of representative words of both document sets.

Second, our dataset is not large and diverse enough. Since the content and writing style in Open Diary is very different from that in New York Times, that makes the results very high accuracy.

In the future, we plan to attack these two limitations by integrating a stop word removal into the preprocessing step and try to obtain data from as many different sources as we can.