# Click Prediction and Preference Ranking of RSS Feeds

December 11, 2009

Steven Wu

## 1 Introduction

RSS (Really Simple Syndication) is a family of data formats used to publish frequently updated works. RSS is often used for temporally relevant information updates. An RSS document consists of a series of entries, each tagged with a title, description, link, date of publication, and sometimes other metadata.

However, it is often the case that a large amount of RSS feeds in aggregate will create considerably more information than the average user will be able to follow. In this case the user will read only those feeds that he or she finds interesting. However, the volume of information involved can sometimes make this search for interesting reading a time-consuming and laborious task.

Several machine learning methods have been applied with some success to text categorization problems, such as Bayesian spam filters for email[1]. Here we explore text categorization algorithms for the purpose of ranking user preferences for RSS entries.

## 2 Preference and Clicks

In our problem we are given a series of RSS feeds and click information and must derive from these the preferences of the user. We are given a set of candidate entries $X = \{X^{(i)}\}$. For each $X^{(i)}$ we define $y^{(i)}$ to correspond to the number of clicks made on $X^{(i)}$. We also store publication time $t^{(i)}$, ie the time at which the entry was produced, and associated timestamp vectors $z^{(i)}$ such that $z_j^{(i)}$ is the time of $j-1$th click on $x^{(i)}$. Hence, $y^{(i)} = |z^{(i)}|$. We will also specifically discuss the time to click $c^{(i)}$, defined as

$$c^{(i)} = \begin{cases} z_0^{(i)} - t^{(i)} & \text{for } |z^{(i)}| > 0, \\ \infty & \text{for } |z^{(i)}| = 0 \end{cases}$$

Each entry $X^{(i)}$ also has a corresponding feed ID $f^{(i)}$ which specifies the specific feed that it belongs to. The importance of this is that different RSS feeds are treated in different ways by the user - a user may check certain feeds more often or be more likely to click entries from it due to topical reasons.

We are attempting to predict preference using click data as an indicator for preference. However, clicks are not a perfect indicator of preference, in that the average user will not have an opportunity to click every article that interests them. It is also the case that even if an article is to be clicked, it will likely not happen for some nonzero amount of time after that article has been published on RSS.

Hence we will make a few assumptions regarding click behavior. First, we assume that there exists some latent variable $I^{(i)}$ tied to $X^{(i)}$ which is an indicator for whether an entry will ever be clicked. We also assume that if an entry $X^{(i)}$, then $c^{(i)}$ is distributed as an exponential. This is to deal with the following difficulty: given some time $t$, if we have some entry $X_i$ where $y_i \geq 1$, then we know both $c_i$ and $I_i$. However, if, conversely, $y_i = 0$, then we have 3 possible explanations. First, the entry may not ever be read, ie $I_i = 0$. Second, the entry may be too old, and the reader may have seen the entry elsewhere, ie $t - t_i$ is too large. Lastly, the reader may eventually read this entry, but not have had the opportunity to yet, ie $t_i + c_i > t$.

We may now formally state that our goal is to determine $P(I^{(i)} = 1)$, i.e. the chance that the user will ever read a given feed entry.

With these assumptions we can find probability $p(y_t^{(i)} > y^{(i)})$ by specifying functions for the rate parameter $\lambda$ of the exponential, $f_\lambda(X)$, and a preference function $f_I(X)$ returning a vector of $\lambda$ and $I$ respectively. We then need to decompose the distribution into a time distribution (exponential) and an binomial indicator. The standard Bayesian technique here would be to construct a generative model for $y^{(i)}$, $z^{(i)}$, and $c^{(i)}$ based on $X^{(i)}$. However, the exact motivations behind RSS preference are not always very clear and often vary from case to case, and so this is not feasible in our case. Hence, it would be ideal to train

a classifier under this model. Naively, we can inject features into our data by creating for each data $X^{(i)}$ a large number of $y_t^{(i)}$ indicating the number of clicks at each time $t$. However, this is not entirely feasible, as we would have to increase an already large dataset by another very large constant factor. Hence, the naive approach fails here.

However, in general we are viewing $y^{(i)}$ and $z_i$ as time series, and attempting to model $y$ and $z$. By MLE, given the rate parameter $\lambda$, we can train a regressor on

$$y^{(i)\prime} = \begin{cases} 1 & \text{for } y^{(i)} \geq 1, \\ \int_t^\infty \text{Exp}(t', \lambda) dt' & \text{for } y < 1 \end{cases}$$

Note that here $y^{(i)\prime} = p(I^{(i)} = 1)$ given our assumptions. In this case we can determine $\lambda$ merely by looking at all entries $X^{(i)}$ that can no longer be clicked (ie are no longer on the RSS feed) and their respective $c^{(i)}$. We can then simply determine $\lambda$ using any number of analytical methods.

With the knowledge of this distribution in mind we can then adapt traditional text categorization algorithms to use the soft labels $y'$.

# 3 Data and Methodology

Our dataset consists of 112,828 RSS entries from a total of 163 unique RSS feeds, of which 2,607 have been clicked. These were collected over a period of three months and represent a cross section of real-life RSS usage.

For evaluation purposes we pick a time $t$ and partition the dataset such that we train on $X_i$ where $z_{i,0} < t$ and test on the remainder. In all the cases we use a modified vector space model where the first dimension of our feature space for $X^{(i)}$ is $f^{(i)}$, and each following dimension corresponds to a word in the corpus. For all tests here we train on the first 500 entries chronologically and then attempt to predict future clicks on entries beyond those 500.

## 3.1 Naive Bayes

A naive Bayes classifier is a special case of Bayesian network applied to the task of classification. Naive Bayes assumes that every feature $X^{(i)}$ is conditionally independent from every other feature given $y$, or, formally,

$$\text{P}(X^{(i)} = X | y' = c) = \prod_i \text{P}(X_i^{(i)} = X_i | y' = c)$$

Using Bayes' rule our hypothesis looks as follows:

$$h(X) = \text{P}(I^{(i)} = 1 | X) = \text{P}(y' = 1 | X) = \frac{\text{P}(I^{(i)} = 1) \prod_i \text{P}(X_i = x_i)}{\prod_i \text{P}(X_i = x_i | C = c_k)}$$

In our case we are attempting to classify based on soft labels $y'$ rather than discrete labels. To account for this when training, for each entry $X^{(i)}$ we consider it instead to be $y^{(i)\prime}$ entries with $y^{(i)\prime} = 1$ and $1 - y^{(i)\prime}$ entries with $y^{(i)\prime} = 0$.

## 3.2 Regularized Logistic Regression

Logistic regression is a conditional probability model in which we assume that the posterior is logistic, ie that

$$p(I^{(i)} | \theta, x_i) = \frac{1}{1 + \exp(-\theta^T x)}$$

Here we can generate $\theta$ by minimizing the negative log-likelihood

$$l(\theta) = -\sum_i \log(1 + \exp(-\theta^T x y'))$$

across our dataset. Note in this case we use the calculated $y'$ rather than $y$. We $l1$-regularize our dataset by using the lasso algorithm, which is known to have good performance with text categorization[2], hence

$$l(\theta)_{lasso} = l(\theta) + \alpha|\theta|$$

where we use Mallows' $C_p$ as $\alpha$. We can then easily solve for the maximizing value of $\theta$ using this function with gradient descent.

### 3.3 Decision Trees and Random Forests

For the purposes of generating decision trees we use Quinlan's C4.5 algorithm [3]. Again, to account for the use of soft labels when maximizing information gain, for each entry $X^{(i)}$ we consider it instead to be $y^{(i)\prime}$ entries with $y^{(i)\prime} = 1$ and $1 - y^{(i)\prime}$ entries with $y^{(i)\prime} = 0$.

We create random forests as described by Breiman et al in [4]. In the random forests technique, we construct each individual decision tree as above, bootstrapping the data in each case, to create the complete ensemble.

### 3.4 Support Vector Machines

The case of SVMs is a unique one since SVMs cannot consider soft labels. Hence we train the SVM only by using those entries in $X$ that have been removed from user access and hence can no longer be clicked, using a Gaussian kernel with Automatic Relevance Determination (ARD). Apart from this pre-processing the training is a reasonably straightforward application of Platt's SMO algorithm on these points [5].

## 4 Results and Conclusions

We run each of these algorithms on the data set as described. The performance of each algorithm can be seen in Table 1.

Perhaps one of the most interesting results here is the unusually good performance of naive Bayes. Support vector machines and random forests have been shown to significantly outperform naive Bayes in [6]. However, here we see that naive Bayes performs only slightly worse than random forests and in fact outperforms the SVM.

We suspect this performance gap is caused by most of the predictive power of our features being in the feed ID $f^{(i)}$ rather than in the actual bag-of-words. This is intuitively sensible, as feeds, like human interests, tend to be restricted to a few topics, and an "interesting" feed will tend to be clicked significantly more often. This is also an explanation of the poor performance of logistic regression: logistic regression cannot account for the feed IDs properly as it can only separate in the feature space, in which a meaningful distinction and ordering does not exist for the feeds - hence, it is forced to operate entirely on the textual information given. SVMs suffer a similar plight. The remaining algorithms that can consider a feature independent of any given space fare much better.

The strong predictive capability of the feed ID intuitively makes sense. Most RSS feeds tend to be focused around one or two topics, as do most user preferences. Hence, if a user clicks on entries from a given feed it is quite likely that he is interested in the topics that feed provides. One can think of the feed ID as a latent variable linked directly to the topic of the feed.

We also note a strong mutability in preferences. Figure 1 shows the ability of three of the classifiers trained on data, and then tested on increasingly more recent data. We note an almost exponential degradation of prediction quality as we increase our prediction horizon. Hence, in order to properly predict clicks we need a dynamic model, or at least one that is frequently re-trained.

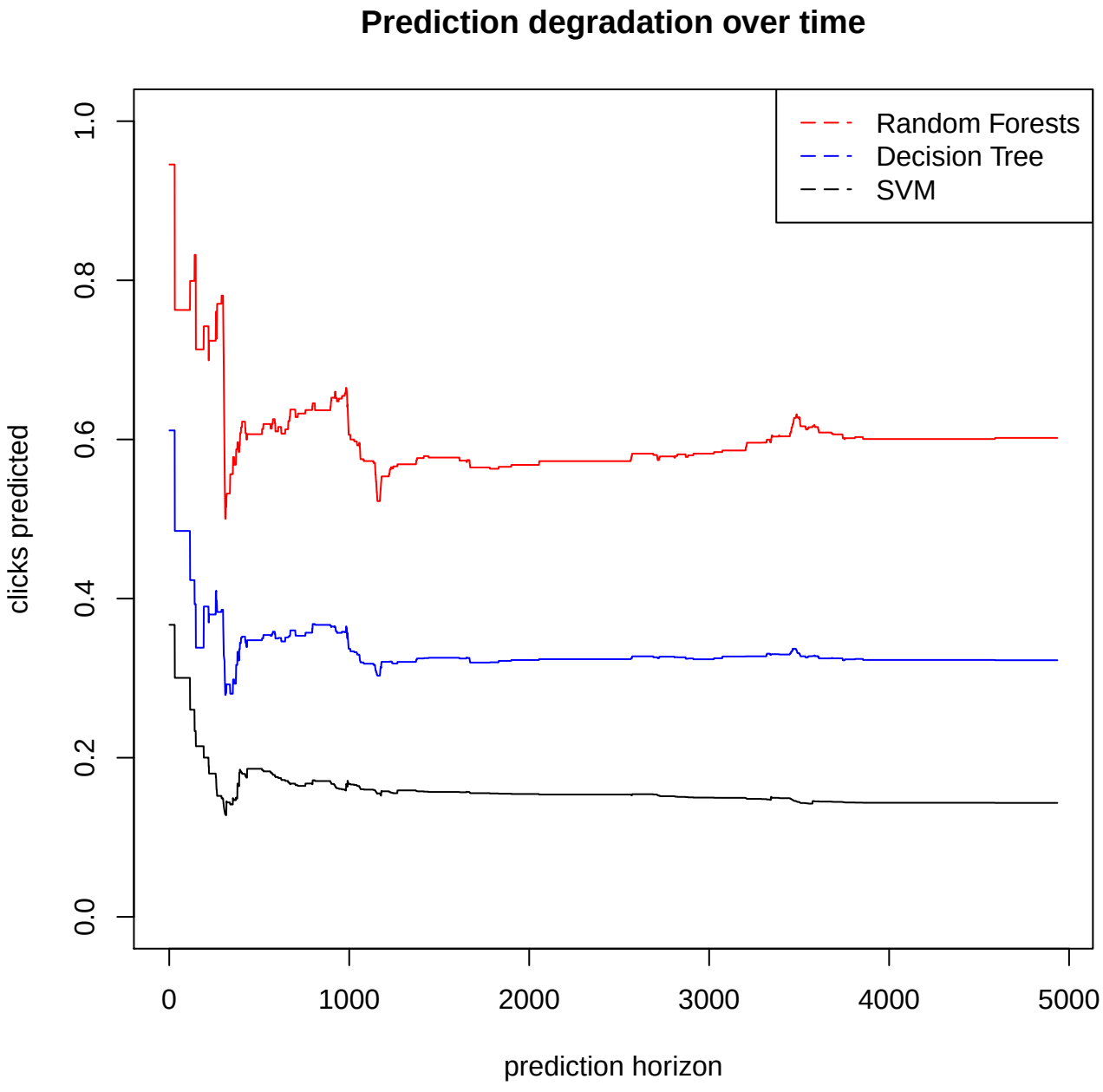


Figure 1: Predictive performance over increasing prediction horizon. Time t

| Algorithm | RMSE |
|---|---|
| Naive Bayes | 0.79 |
| Regularized Logistic Regression | 0.92 |
| Decision Trees | 0.86 |
| Random Forests | 0.76 |
| SVM | 0.82 |

Table 1: RMSE on testing set for the various classification algorithms.

# 5 Further Directions

As it seems the most performant feature we observed was the feed ID, which seems to derive its predictive power from being directly linked to topic, the next logical step seems to be topic models. We plan on exploring Latent Dirichlet Allocation and the hierarchical version thereof in order to create topic models with RSS feeds, which will allow us to classify with potentially greater accuracy than before. Other possible approaches include collaborative filtering; however, these require a volume of data that we, at this moment, do not have access to.

# 6 References


1. Sahami, M., Dumais, S., Heckerman, D. and Horvitz, E. (1998) A Bayesian Approach to Filtering Junk E-mail. *AAAI Workshop on Learning for Text Categorization.*

2. Genkin, A., Lewis, D. and Madigan, D. (2004) Sparse Logistic Regression for Text Categorization.

3. Quinlan, J. (1993) *C4.5: Programs for Machine Learning.* Morgan Kaufmann Publishers.

4. Breiman, L. (2001). Random Forests. *Machine Learning.*

5. Platt, J. (1999). Fast training of support vector machines using sequential minimal optimization. *Technical Report MSR-TR-98-14.*

6. Caruana, R. and Miculescu-Mizil, A. (2006) *Proceedings of the 23rd international conference on Machine learning*