Pulse Project: User-Interest-based News Prediction

Yinan Na Jinchao Ye Yijun Liu

Abstract

Pulse is a news recommendation app available on both iPhones and android phones. Predicting news of users' interest according to their reading history has always been a hot topic. In this project, we used tf-idf vector and Logistic Regression to predict news of users' interest on Pulse. The feed of news has been proved to be a key factor for predicting news. Special Measures have been taken to handle unbalanced training data. We use word stemming and other techniques to reduce the dimension of tf-idf vector. We also compared Logistic Regression with Support Vector Machine. Moreover, instead of treating each user independently, we also tried unsupervised learning methods such as K-Means on users.

1. Introduction

1.1. Motivation

Today, many users love to read news from their mobile devices, where they can easily pick up news stories recommended by app softwares like Pulse. In such applications, the program predicts what kind of story the user's likely to read, by applying various machine learning algorithms on the dataset. The recommendation system is like Netflix Challenge [2]. Our project aims at developing and improving such prediction algorithm, to provide an as accurate as possible result.

1.2. Dataset

We use the Pulse data in August (31 days) as our dataset. There are three files, stories.log, user_story_read.log and user_story_clickthroughs.log. In stories.log, each story is stored in the format <story URL, story title, feed URL, feed title, timestamp>. There are total 7349312 raw entries in this file. The read.log is stored in the format <user, story URL, story title, feed URL, feed title, timestamp>. There are 2466556 reading records in this file. The click.log is stored in the same format as read.log. There are 92620 clicking records in this file.

1.3. Problem Definition

For each user, we build a classifier and evaluate it over the time series:
Train on day 1, and test on day 2, 3,…, 31
Train on day 1,2, and test on day 3, 4, 5,..., 31
…

The classifier predicts whether the user will read or click a certain story in the following days. We employ F1 score as criteria for evaluation.

2. Approach

2.1. Data Pre-processing

We removed the malformed or duplicated data in stories.log. By "malformed" data, we mean those entries which lack parts such as story title, story URL, feed title, feed URL or timestamp. By 'duplicated' data, we mean those entries which are exactly the same except for the timestamp. In this case, we only reserve the one with the earliest timestamp. There are 333458 stories left out of 7349312 raw entries after the processing. We also adjust the other two files according to stories.log.

2.2. Baseline – tf-idf vector and logistic regression

The tf-idf [1] is a weight used to evaluate how important a word is in a document. It is often used in information retrieval. More specifically, let t denote a word, d denote a document and D is the set of documents. Then

$$idf(t) = \log\frac{|D|}{|\{d: t \in d\}|}$$

$$tf-idf(t,d) = tf(t,d) * idf(t)$$

Here $tf(t,d)$ is the term frequency in the documents. The higher the score is, the more important the word is in the document. Before we use word stemming and other techniques, the tf-idf vector is an 119836-dimension vector. When computing the tf-idf vector, we ignore the non-English characters.

As taught in class, Logistic Regression is a common to predict the probability of an event when no prior information is available.

We compute the tf-idf vector for each story. Then for each user, on each day, we train a classifier for predicting whether a certain story will be viewed or clicked by the user.

2.3. Handling the unbalanced training data

One tricky part of the problem is that during the training process, there are much more negative samples than positive samples, as shown in figure 1. If we use these data directly, the classifier we get might judge every test sample as negative.

We employ an iteration scheme to solve this problem. For the first iteration, we first randomly pick up the same number of negative samples as positive samples and feed them to the logistic regression. Then we use the classifier to test the whole training data and pick up top 10 hard negative examples to replace 5 original negative samples. Then we feed the newly generated negative samples along with positive samples to logistic regression. We did iteration for 100 times and get the final classifier. The idea is similar to Boosting [3]. We did not replace all the negative samples at one time because that will make the classifier change dramatically and not converge, just as figure 3 shows. We replace 5 negative samples each time and this makes the decision boundary moves gradually towards the ideal place, just as figure 2 shows. This handling increases the F1 score.

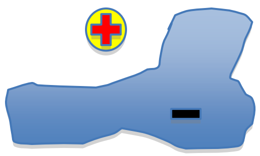
Figure 1 Unbalanced Data

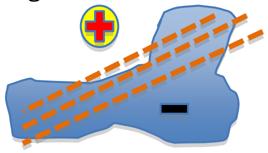
Figure 2 Boundary Moving towards Ideal place

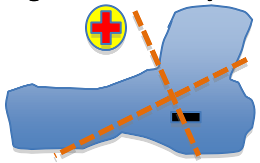
Figure 3 Boundary oscillate dramatically

2.4. Word stemming and dimension reduction

Use the 119836-dimension tf-idf vector as feature not only seriously lowers the computation speed, but also causes the problem of over-fitting.

We first remove the stop words such as “a”, “the”, “here”, “there”, etc.

Word stemming is a common method of reducing the dimension of tf-idf vector. For example, it will map the words “process”, “processes”, “processor”, “processing”, “processed”, procession, to the same word “process”.

However, after we have implemented word stemming, the dimension of tf-idf vector is still at a relatively high level. So we tried Latent Dirichlet Allocation (LDA). However, since the packages which implemented LDA are all too large and required 4GB memory, we find it hard to employ them in our own package.

We use our own method instead. We find a vocabulary table of 8000 words. These words are frequently used and are sufficient for most uses. We first removed the words that are not in the 8000 words vocabulary. Then we did word stemming on the 8000 words and got 5840 words instead. In the meantime, we also record the word hierarchy. The idea is similar to WordNet [4]. These dimension reduction methods increase the F1 score.

2.5. Removing false positives using feed

Despite that we have incorporate feed URL and feed title when we compute tf-idf vector, the feed title can be used more efficiently as a cue to remove false positives.

We have noticed that some readers only read news from certain feeds, such as “USA Today”, “Hacker News”. This is easy to explain just as most people usually buy clothes from certain stores. Therefore, for each user, we maintain a set F, which records the feeds the user have read news from. During testing process, we directly label those stories of which feed title are not in F as negative. This means we assume that the user only read news from feeds in F. This is proved to remove a lot of false positives while only slightly lower the recall rate. Therefore it improves the performance.

2.6. Support Vector Machine

We also use Support Vector Machine (SVM) as a comparison method to Logistic Regression. As taught in class, Support Vector Machine finds line with the largest margin between positive samples and negative samples. SVM is usually performs better than logistic regression, this is also confirmed in our experiment.

2.7. User Clustering using K-means

As for unsupervised learning algorithm, we tried using K-Means to cluster 500 users to K clusters. In our experiment K = 10. The vectors that are used for clustering are 333458 dimensional binary vectors. The jth component is 1 if the user has read the jth story before certain day otherwise it is 0.

Once we have clustered the 500 hundreds users, we can assign each of the remaining user to one of the K centers according to their previous reading history. Then on a new day, for a user s belonging to cluster L(s), we recommend the top N stories most read by other L(s) users on that day for user s.

3. Experiment

We use the dataset described in section 1.2 and 2.1. The statistics of the processed dataset is shown as Table 1:

Table 1 Statistics of Processed Data

Number of stories in 31 days	333458

Number of users	1094
Number of read times	2292520
Number of click times	87354

We find that many users seldom click through a story on many days and it is very difficult to predict whether a user will click through a story, so we do not incorporate the third file, user_story_clickthroughs.log, i.e. we do not incorporate the clicking data.

Also, we find that in the 1094 users, many users neither reads or clicks through any story. We pick up the most active 200 users for our experiment.

As stated in section 1.3, for each user, on each day j, we train on the data on the first j days and predict whether or not the user will click the new stories in the following 31-j days. So for each user on each day, we can get a precision score, recall score, as well as F1 score. Here

$$F1 = \frac{2PR}{P+R}.$$

We have implemented and compared five methods. The five methods are explained as following:

LR: this simply uses tf-idf vector and logistic regression. This does not employ handling of unbalanced data. During training, the ratio between negative samples and positive samples are 5:1. The negative samples are randomly picked. This is the baseline.

LR + iteration: In addition to the baseline, this method handles the unbalanced data use iteration method described in section 2.3.

LR + feed: In addition to the baseline, this method uses feed titles as cues to remove false positives, as described in 2.5.

LR + Stemmer + feed: This method uses logistic regression and tf-idf vectors as well, but the tf-idf vectors are reduced to low dimension using word stemming and other methods described in section 2.4. It also uses feed as a cue to remove false positives.

SVM + feed: This method uses SVM as learning algorithm and tf-idf vector as feature. Besides, it also uses feed title to remove false positives.

We used liblinear and libSVM packages for logistic regression and SVM.

Table 2 Precision, Recall and F1 Comparison

	F1	Precision	Recall
LR	0.2678	0.159	0.9346
LR+Feed	0.4688	0.3402	0.9029
LR+Stemmer+Feed	0.4619	0.3201	0.8981
LR+Iteration	0.4168	0.3001	0.738
SVM+Feed	0.5066	0.3694	0.8712

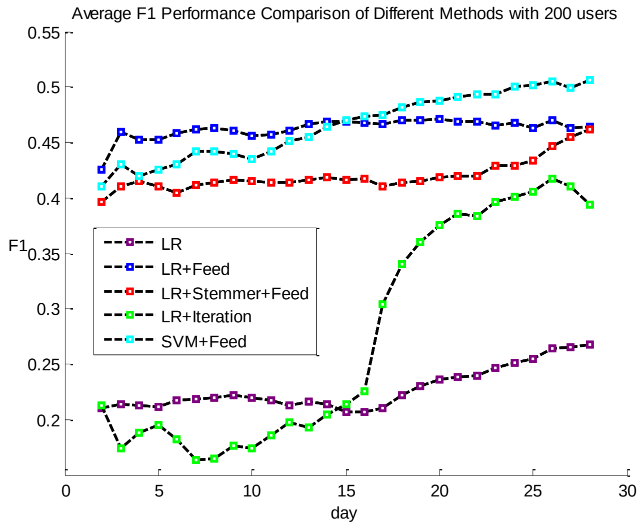

Figure 4 Average F1 for 200 users

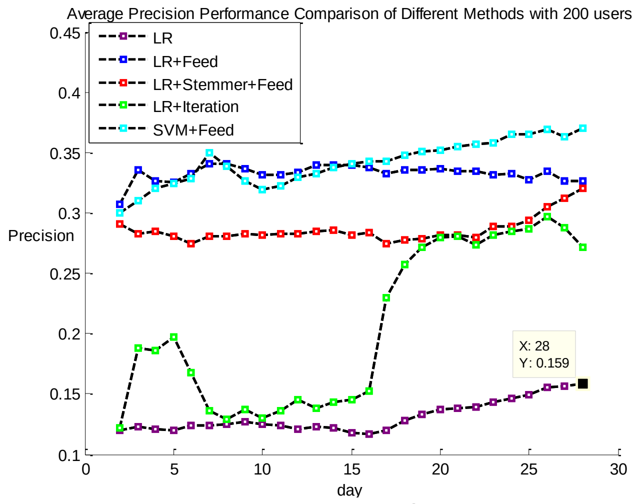

Figure 5 Average Precision for 200 users

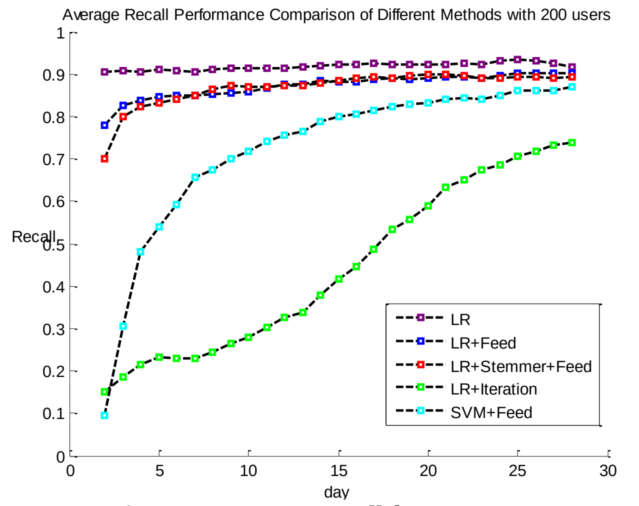

Figure 6 Average Recall for 200 users

Table 2 records the average precision, recall and F1 for 200 users. Despite it is the average precision of different users, it pick up the peak value during 31 days.

From the F1 curves, we can see that SVM + Feed out performs other methods. And methods with feed outperform those without feed. LR with iteration outperforms that without iteration.

From the Precision curves, methods with feed have better precisions than those without using feeds. This is because by using feed, we can remove a lot of false positives.

From the Recall curves, we can see that baseline method has the highest recall. It is because we only pick up randomly picks up a small number of negative samples during training and the classifier therefore is easier to tends to categorize the training data as positive. So it has high recall but extremely low precision, and hence the lowest F1 score. When we add iteration during the training process to handle the unbalanced data, the recall drops while the precision rises and the overall performance F1 rises. The SVM + feed method balances precision and recall well and therefore has the highest F1 score.

Another point is that different users have really different performances. For some user as shown in figure 7, the F1 score is really high. Of course, there are also users whose behavior is hard to predict.

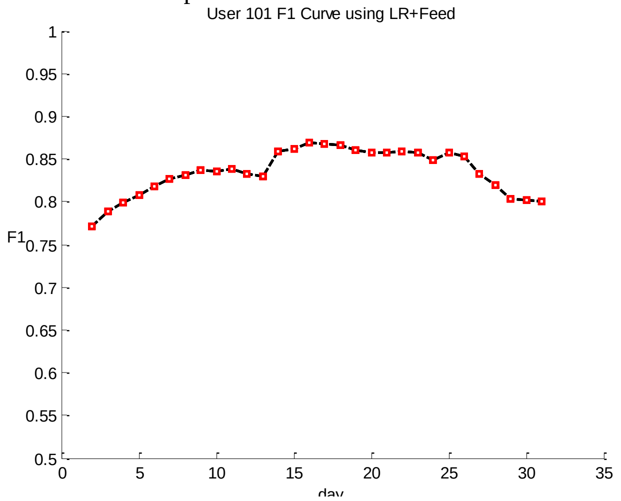

Figure 7 F1 Curve for user 101 using LR+Feed

As for user clustering, our implementation is problematic because it clusters 490 users into one cluster while other clusters each have only 1 or 2 users. We are still investigating the problem.

4. Conclusion

The above experiments have shown tf-idf vector is an effective feature for information retrieval or text mining. More importantly, Logistic Regression and SVM have both been proved to be effective for large-scale machine learning projects. SVM is better than Logistic Regression in accuracy while it needs a little more computing time during training.

In order to acquire good performance, we need to employ feed title as an important cue to remove false positives. The dimension reduction of tf-idf vector, such as using word stemmer or LDA, is also critical to this problem as it not only speed up the computing process, but also avoid over-fitting. As for unbalanced training data, we need to iteratively pick up equal number of negative samples and positive samples and train on them. The rule for picking up these samples is similar to the idea of Boosting.

5. Future Work

Currently we have done the dimension reduction via word stemming and lookup-tables, which helped us recognize all those common English words and named entities and eliminate those non-sensical strings. In the future, we want to try LDA and compare with our own method of dimension reduction. Principle Component Analysis (PCA) might also be used to further reduce the dimension of tf-idf vector. We can also try Named Entity Recognition to reduce the dimension.

Despite that the clicking data is really sparse, we may also use it to predict whether a certain story will be read or viewed by a user. In fact, we can use weighted logistic regression and assign higher weight to clicked-through story. This is because people click through a story because they are more interested in this story than those who are only viewed.

Our implementation of user clustering using K-Means seems to have problems. So we still needs to investigate why we cluster so 490 users to 1 cluster while other clusters only have 1 or 2 users each. Moreover, we might also use other unsupervised learning algorithms such as collaborative filtering.

6. Acknowledgement

We would like to express our gratitude to all the CS229 teaching staff who gave us the possibility to complete this project. We want to thank Prof Andrew Ng for all the ineffable lectures, along with the meticulously articulated course notes.

We want to thank Dr Richard Socher for his altruistic and comprehensive instructions in this project. Without his trenchant advices, the progress of our work could have ceased to move at the very beginning. All his instrumental advices greatly incited our minds and helped us bring about this final project.

References

[1] Karen Spärck Jones, A statistical interpretation of term specificity and its application in retrieval. Journal of Documentation 28 (1): 11–21.

[2] Robert M. Bell, Yehuda Koren and Chris Volinsky, The BellKor solution to the Netflix Prize.

[3] Yoav Freund, Robert E. Schapire, A Decision-Theoretic Generalization of on-Line Learning and an Application to Boosting. Journal of Computer and System Sciences 55, 119139.

[4] George A. Miller, WordNet: a lexical database for English, Communications of the ACM, Volume 38 Issue 11, Nov. 1995.