Faceloook: Learning user preferences on Facebook feed

Hsiang-Sheng Liang *Jie-Wei Wu*

Department of Electrical Engineering
National Taiwan University
{b97901125, b97901084}@ntu.edu.tw

ABSTRACT

We present Faceloook, a browser extension that learns user preference on her/his own Facebook feed. Faceloook uses click events on each posts to distinguish the interesting posts from the others, collecting labels without user's cognitive effort. Naive Bayes classifier is used to create the user preference model out of contextual features of the posts and the click event labels. 5-fold cross validation is performed to evaluate the system. The entire project is available on the Internet[1].

INTRODUCTION

Browsing the overwhelming feed on Facebook is time-consuming, not to mention searching a specific post on the feed. However, the post that a user really interested in is much less than those the user have read. Users often find themselves scrolling down the their news feed merely to pick up one or two posts they really interested in. This is still true even with the "Top stories" version of the news feed, which shows only feeds according to the affinity score and the social events occurred on the post [1].

We believes that a user preference model can do better on choosing what a user is really interested in. User interactions and the contextual cues when browsing the feed can help us conclude whether a post is interesting or not. Combined with the context of the post as the machine-learning feature, a user preference model on Facebook feed can be constructed.

Faceloook is a Google Chrome extension that records user clicks on the user's Facebook news feed. A simple click event can be any interaction between the post and the user, such as "like" and "reply". When such interaction exists, we suppose that the user is interested in the post. The click event data and the post context is used to train a Naive Bayes classifier, which is the user preference model that distinguishes interesting posts from others.

As an application of the model, the browser extension also provides a query interface, enabling search on Facebook feeds. The results are sorted according to the classifier output, helping users to find desired posts quickly.

[1] https://github.com/MrOrz/faceloook

Figure 1: Screenshot of Faceloook query interface. The figure shows the posts classified as "interesting" to the user, guessing what post the user might want to search even before he/she enters a search term. The order is determined using the trained user preference model.

RELATED WORK

Webb et al. [2] identifies several challenges in user modeling, including the need for large datasets and labeled data, concept drift and computational complexity. Michalski et al. [3] discussed algorithms for learning and revising user profiles that can determine which websites on a given topic would be interesting to a user. They used a naive Bayesian classifier for this task, and demonstrated that it can incrementally learn profiles from user feedback on the interestingness of Web sites. In an experimental evaluation, They compares the Bayesian classifier to computationally more intensive alternatives, and show that it performs at least as well as these approaches throughout a range of different domains and empirically analyzing the effects of providing the classifier with background knowledge in form of user defined profiles and examine the use of lexical knowledge for feature selection.

Facebook once had a function called "news feed preferences", which allows users to adjust which types of story to show up in her / his news feed. However, it requires user intervention and cognitive effort to tune the preference settings. The function is no longer available since 2009.

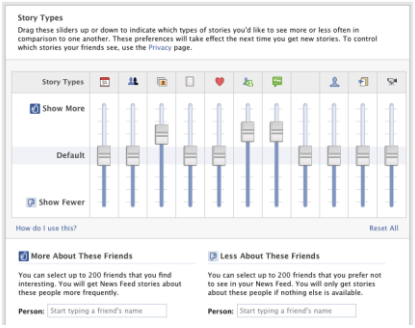

Figure 2: News feed preferences of Facebook.[2]

DESIGN AND IMPLEMENTATION

Faceloook is implemented as a Google Chrome extension. It collects user interaction with the Facebook feed. The interaction information is combined with contextual information of the posts, which consist of the training feature and the interesting / not-interesting label. The feature is processed by word segmentation system, before being modeled by the Naive Bayes algorithm. The extension also provides a feedback mechanism to let user actually see the classification result, and the ability to alter the label. Finally, a query interface is built, with the user preference model in mind, enabling search functionality over the user's news feed. The system block diagram is shown in figure 3.

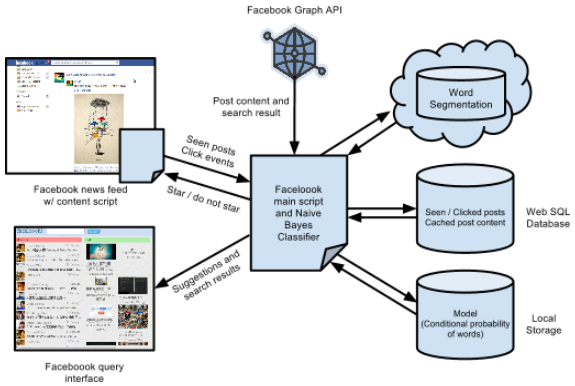

Figure 3: Block Diagram for Faceloook, the Google Chrome extension.

User Interaction Recording

The very first thing the extension achieve is to collect user interaction with the posts in the feed. The browser extension content script stores Facebook object ID of all posts into the Web SQL Database, but only those that were once appeared in the screen has a "updated_at" timestamp, indicating the last interaction time. The "updated_at" timestamp is important in two ways. First, only "seen" posts are meaningful to user preference modeling, since we cannot say the user is not interested in an unseen post just because she/he does not see it yet. The existance of "updated_at" timestamp marks the post as "seen", or "once appeared in screen". We use only "seen" posts to train the model. Secondly, the timestamp can be used to impose different weights on the classifier output, which is explained in detail in the "query interface" section below.

Another interaction the extension concerns is the "click" event mentioned before. Any click event occured on the area of a post (figure 4) will mark the post as "clicked" in the Web SQL Database. The "updated_at" timestamp is updated when a post is clicked.

Figure 4: The purple shaded area is the area of a feed. Any mouse click action, including like, comment, or drag-selecting words within will mark the post as "clicked".

Feature Extraction

From the context of the post, we extacted a couple of features that affect a user is interested in a post or not. Table 1 lists the features that we considered to be the typical features that really matters.

message	The content of the post.
link	The URL of the link included in the post.
caption	The title of the link.
description	The content of the link.
from	The author of this post, in Facebook Object ID.
type	Could be one of "status", "photo", "video", "link", or "checkin".
group	The group ID this post is posted in.

Table 1: Features of a post on the feed.

[2] http://www.chewie.co.uk/facebook/facebook-news-feed-preferences-no-longer-work/

We used MMSEG[4], a well-known Chinese tokenizer, to segment the post contents into words. Since word segmentation is not the main focus of this work, we directly adopted a ruby implementation [3] of MMSEG algorithm for the convenience of implementation. The dictionary file is from the Chewing input method. The word segmentation system is deployed as a web service[4] for the browser extension to access.

Machine Learning

How Faceloook trains the model is basically very similar to training a spam mail classifier. All tokenized text features like caption, description, and even links are concatenated and is marked with label "interesting" or "not-interesting" according to the presence of click event. Features that identifies a specific user or group like "from" and "group" are represented by their Facebook object ID. Post types are transformed into tokens that does not appear in our daily life. The types are prefixed by the string "TYPE", resulting in post type tokens like "TYPEstatus" or "TYPElink".

Faceloook uses a third-party classifier library, brain.js [5] , to train and test the Naive Bayes model. We modified the classifier library to output a score, which is the probability the user is interested in the post. The training procedure starts in background when the user opens Google Chrome. Only posts that were marked "untrained" were processed at that time. When user visits Facebook, newer posts with "untrained" mark will enter the Web SQL Database, awaiting click events. These posts will be trained after the restart of Google Chrome.

Figure 5: The star in each post indicates the classification result. A yellow star means this post is classified as "interesting", or the user has manually marked the post as "interesting".

[3] http://rmmseg.rubyforge.org/

[4] A Ruby on Rails Application is created specifically for the word segmentation. The application is deployed on Heroku (http://www.heroku.com) and has a public domain name.

User Feedback

Faceloook inserts a star-shaped mark (figure 5) into each post in Facebook feed. The color of the star reflects the classification result. The star also acts as a toggle switch, enabling users to manually mark an individual post as interesting or not-interesting. When user clicks on the star, the label of the specific post is altered, but only if the post is still not trained yet.

To insert the star-shaped mark, Faceloook must classify each post on the current feed first. We first obtain the context of each feed using Facebook Graph API, given the object IDs recorded by the browser extension content scripts. The context is then segmented, and classified with the current model. An empirical threshold of 0.7 is chosen for the classifier. The word-segmented results are cached in database to speed up the training procedure.

Query Interface

Figure 1 shows the query interface of Faceloook, as an application of the user preference model. Clicking on a post item leads the user to the original post. The feed posts are categorized and sorted by the probability the user is interested in each post. The probability is weighted by the age of the post. More specifically, the age is determined with the following formula:

$$\exp\left(-\text{updated time}\right)$$

Where the "updated_time" denotes the time the post is seen or clicked by the user.

At first, the latest 50 posts are presented, sorted using the method mentioned above. These posts are suggestions made by Faceloook, predicting that these posts are most likely to be what the user might want to search for, before the actual search term input. After the user enters a search term and submits the search form, queries will be made to the Facebook Graph API. The search results are segmented and classified, as what Faceloook did when putting stars on the feed messages. Finally, the search result is categorized with its post type, and is sorted by the weighted probability.

EVALUATION

We created several agents to emulate user clicking behavior. After the agents label all the posts cached in the Web SQL database, a 5-fold cross validation is used to generate the score of the user preference model. The posts in the Web SQL database are real-world Facebook posts collected in one month from 2 users, constituting 6200 posts as our test data. We split the test data into 2 groups. The first contains 1200 posts, which is the number of the posts an user may read in a week. The other group has the rest of the posts, emulating a longer period of data collection.

The agents consists of simple rules to determine if it is interested in the post. The rules involve seeking for specific keywords in the message attribute, checking if the post is from a specific group or user, or if the content belongs to a specific type. The agent we used for the cross validation

deterministically labels 85% of the posts as not-insteresting. The percentage is empirically derived from the actual click data collected from the 2 users; that is, they clicked 15% of the posts they had seen on their news feed.

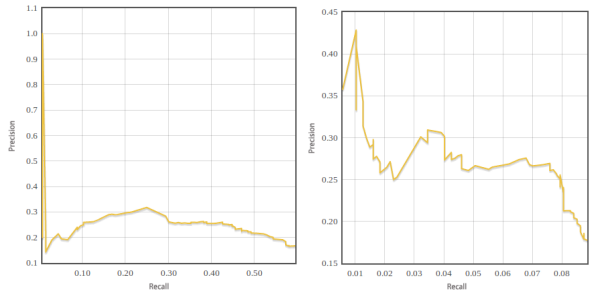

Figure 6: The PR curve of 1200-post group (left) and the 5000-post group (right).

Figure 6 shows the PR curve of the two groups of the test data. The precision and recall is far from desirable in the both groups. As the threshold variates, the average precision is approximately 0.3 and the recall rarely exceeds 0.5. Recall rate of 5000-post group looks particularly awful. The longer data collection period does not seem to improve the performance at all.

To compare the classifier performance with the "always-guess-0" baseline classifier, figure 7 illustrates the accuracy-to-threshold graph of the both group. With 85% of test data is marked as not-interesting as the ground truth, the baseline classifier accuracy should be 0.85. It turns out that the overall performance of the Naive Bayes classifier does not improve much from the baseline. The optimal threshold giving the best accuracy is above 0.9 for both cases, which is no different from the "always-guess-0" classifier.

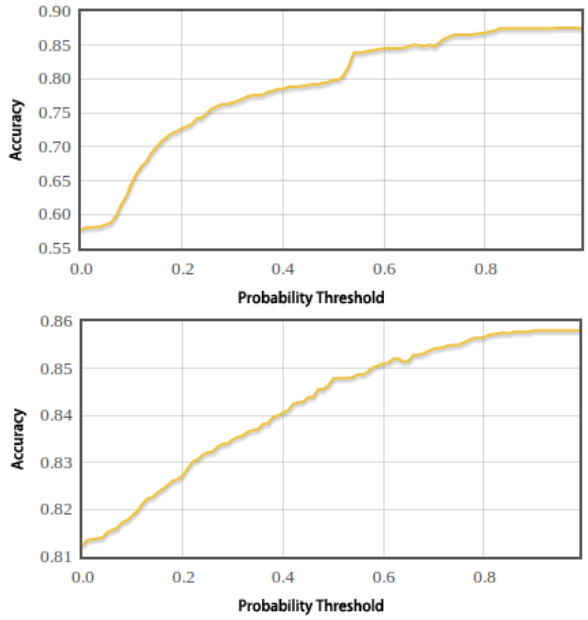

Figure 7: The accuracy-to-threshold graph of 1200-post group (top) and the 5000-post group (bottom). The classifier output of a post must be larger than the threshold in order to be marked as interesting.

CONCLUSIONS AND FUTURE WORK

We implemented a system to create a user preference model using the contents from Facebook content and the user click data. However, the performance of the trained classifier is far from useful, leaving a great space of improvement. Some directions of improvement are listed below:

- When combined with other classifiers like SVM [6] and a language model, the classifer may have better performance compared to merely using the Naive Bayes Algorithm
- Filter out common words in Chinese language before training Naive Bayes classifier. This is a common way to rule out irrelevant words that has little impact on whether the post is interesting or not.
- Another one need to be improved is the speed. In the front-end, it would renew the post while all search data received and finish all probability calculating. The transport of data depends on the web speed.

REFERENCES

1. http://smartblogs.com/social-media/2010/10/14/how-to-make-your-facebook-content-top-news/
2. Webb, G. I., Pazzani, M. J., & Billsus, D. (2001). Machine Learning for User Modeling. *User Modeling and User-Adapted Interaction* 2001, (1978), 19-29.
3. S.Michalski, R., & Wnek, J. (1997). Learning and revising user profiles: The identification of interesting web sites. *Machine Learning - Special issue on multistrategy learning*, 331, 313-331.
4. http://technology.chtsai.org/mmseg/
5. https://github.com/harthur/brain
6. P.Yeh et al. 結合 SVM 與 Naïve Bayes 演算法防堵垃圾郵件的研究. 2007