CS229 Final Project: Predicting News Preferences

Rory MacQueen, Deniz Kaharamaner, Gil Shotan

December 16th 2011

Abstract

Our goal was to predict news stories that a user would read based on his or her reading history. We came up with a set of features for each story, including tf-idf values of the words in the title, the feed from which the story originated, and a measure of how many similar users read that story. Finally we weighted training stories more heavily if a user both read and clicked through that story. We fed this matrix of features into a linear Kernel SVM[1] and achieved an F1 score[2] of 0.45. While we consider the results to be satisfactory, we suspect the main obstacle preventing us from achieving higher accuracy was our inability to accurately determine which stories a user was exposed to, and, therefore, to determine whether a 'non-read' was because she was not interested in the story or simply because she did not see it.

Introduction

Mobile devices have revolutionized the way people consume news. Over the past three years, the proportion of web time spent reading news has doubled[4]. Moreover, as of 2010, people spend the same amount of time on their mobile devices as they do reading newspapers and magazines combined[4]. This digitization of news consumption opens up an opportunity for machine learning algorithms to improve users' reading experience.

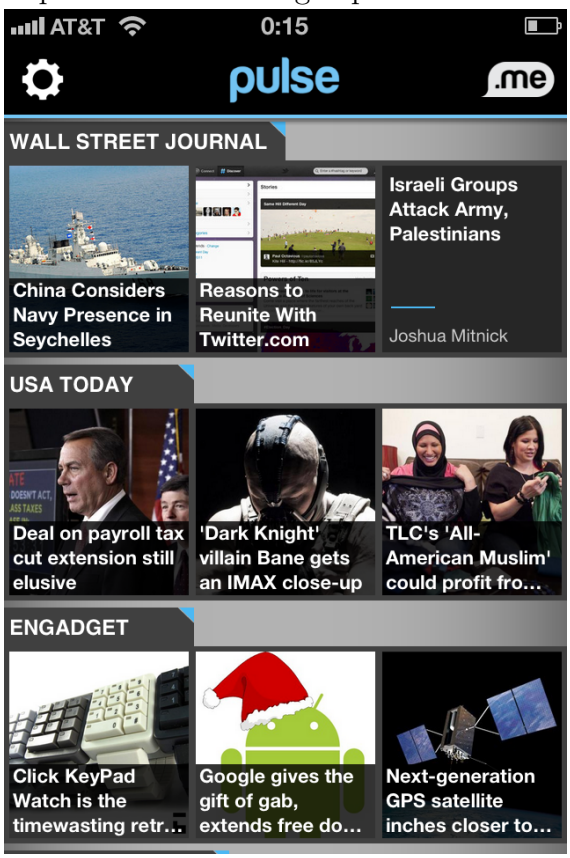

Figure 1: Screenshot of the pulse application

This project applies a machine learning algorithm to predict which stories users will read based on each user's reading history. The ultimate goal is to to tailor a news feed to a particular user's

interest based on his or her past reading habits.

Method

Our data set was three-fold. First, we had a file consisting of all stories available for the month of August 2011. Second, we had a file containing 'reads', showing which stories were read by each of our 1000 users. Third, we had a file showing which stories were clicked through by each of our 1000 users. One of our biggest challenges was taking this large data set, and determining which stories each user was actually exposed to. The two factors determining exposure were timestamps and feeds. We decided that it is reasonable to assume that any day on which a user had no time-stamped 'read' stories indicated a day on which a user did not even log in; hence that user was not exposed to any of that day's stories. We therefore did not include those stories in that particular user's training matrix. Furthermore, we used one scan through the whole data set to determine which feed (e.g. Wall Street Journal, TechCrunch etc.) each user was subscribed to. This allowed us to exclude from the user's matrix any stories from other feeds. We used a set of 5000 stories and 500 users as a working development set. We evaluated our performance iteratively - by increasing our training set one day at a time. At each iteration we attempted to predict the stories read by each user on the following day. Concretely, we started by training on day 1, and tested on day 2; then trained on days 1, 2 and tested on day 3 etc. Eventually, we tested our final system on the other set of 500 users and 50,000 stories. (we did not include all stories due to long duration of training).

Algorithm

We used a linear kernel SVM algorithm[3] to classify our data into reads and non-reads for a given user. To account for outliers and the problem of non-linearly separable data, we use $\mathcal{L}_1$ regularization on the algorithm. We ran the algorithm several times on development data, varying the parameter C, which controls the cost to be incurred for having examples with functional margin less than 1. We found that value of 100 for C was optimal. We also experimented with different weights for the categories, read and non-read, represented in the algorithm as 1 and 0 respectively. A higher weight for a given category tells the algorithm that it is more important that we get the classification of this category correct, even if it might mean classifying some of the other category's points incorrectly. We assigned a higher weight (10) to the reads category, since, intuitively in this scenario, it seems to be more important to have a higher recall at the expense of precision.

NLP

We started by representing a story only using the term frequency and inverse document frequency of the words comprising its title.

$$\text{tf-idf} = \text{tf(t,d)} \times \log \frac{|D|}{|\{d : t \in d\}|}$$

where $|D|$ is the number of documents in our corpus We chose a set of 10,000 tokens to comprise our initial feature space based on their frequency in more than 200,000 stories. As with most natural language corpi, the words in a data set of news stories are distributed approximately according to Zipf's law[5] that is, the frequency

of any word is inversely proportional to its rank in the frequency table. In other words, the most frequent word will occur approximately twice as often as the second most frequent word, three times as often as the third most frequent word, and so on. Furthermore, we excluded 'stop words' from our vocabulary since they have little contextual significance. We also applied Porter's Stemmer to reduce similar words to their common morphological root by applying linguistic rules. Hence the words 'invade', 'invading' and 'invaded' will all be mapped to a common root, namely 'invade', which is what we eventually stored in our training matrix. After this initial phase of pre-processing, we chose the most frequent roots as our set of tokens. We ended up with a very sparse training matrix, as each story contains only several words, and only a fraction of which appeared in our dictionary. Our preliminary results revealed a high bias in our hypotheses, which motivated us to seek other features with higher predictive value.

Incorporating Feeds

We hypothesized that not all feeds are treated equally by a given user. Most users are subscribed to over 20 feeds, whereas an individual user can only view 3 feeds at any given moment on his iPhone screen. An inverse correlation likely exists between the probability that a story will be viewed and the amount of scrolling necessary to reach it. Furthermore, we hypothesized that the ability of a particular feed to match a user's preference may vary significantly. Some feeds may produce stories that match a user's preference more frequently than others. We therefore decided to include the feeds of a story as a feature. A scan of the data confirmed our hypothesis that indeed, some feeds are more popular than others for a given user.

Clustering

Another method we employed to gather insight into a user's preferences was to view the reading patterns of similar users. Our conjecture was that if the reading patterns of a group of users has been similar in the past, they are likely to be similar in the future as well. Intuitively, it is easy to believe that a certain group of users is more interested in sports, others in politics, etc. To test our conjecture we use Principal Component Analysis[6]. We defined the preferences matrix $P \in \mathbb{R}^{m \times n}$, where m is the number users and n is the number of stories we used for this purpose, where each component of P contained one of 3 values:

$$P_{ij} = \begin{cases} 0 & \text{if user } i \text{ did not read story } j \\ 1 & \text{if user } i \text{ read story } j \\ 2 & \text{if user } i \text{ clicked on story } j \end{cases}$$

We implemented PCA on the matrix P to map the high-dimensional user vectors to a three-dimensional space that we could visualize (see figure 2). This visualization suggested that users fall into 3 groups, or clusters. We implemented kMeans[7] clustering on our development users. Each cluster was represented as a vector in $\mathbb{R}^n$, which was computed as

$$c(q)_j = \sum_i P_{ij} \mathbb{I}\{\text{user } i \text{ belongs to cluster } q\}$$

and where $c(q)_j$ is the number of users in cluster q who read story j. During testing, we had to assign each of our new test users to one of these three pre-computed

clusters. Each user was represented by a vector $P_i \in \mathbb{R}^n$, which is a row in the matrix P. We assigned each user to the cluster which had the smallest Euclidean distance from its center to the user's vector.

$$k_i = \arg\min_q ||P_i - c(q)||$$

where k_i is the cluster assignment for the user i. Euclidean distance ended up being a more accurate measure of similarity between a user and a cluster than cosine distance; cross-validation on the training data showed that Euclidean distance method correctly assigned users with more than 90% accuracy. Having assigned a test user to a cluster, we were now able to add to each story of this user's feature matrix an additional feature, whose value was the number of users in the same cluster who read that story. The addition of this clustering feature improved results by approximately 0.05.

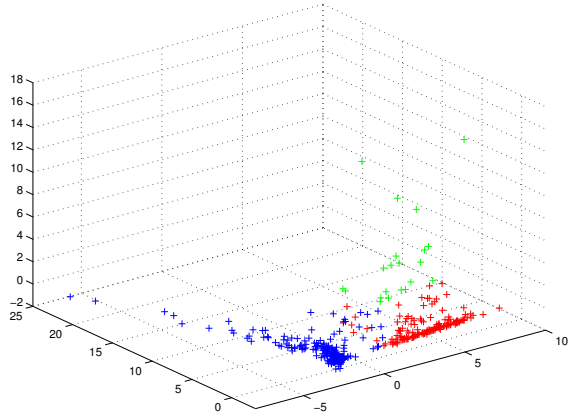

Figure 2: Principal Component Analysis showing 3-dimensional representation of users' preferences, colored-coded by cluster assignments

Incorporating Clicks

A 'click through' event means that a user not only read but also opened up the full article in its original website, thereby indicating that she has an even greater interest in this story. To account for this information, we want to tell the learning algorithm that the words in this title have even greater predictive value of a user's interest, i.e we want to amplify the tf-idf values of these words in all future stories. To accomplish this, we take each word in the title and multiply its corresponding 'feature column' by a given 'click-factor'. This gives more weight to future stories that contain these words in their title. Experimenting with different values, we found that a click factor of 1.5 was optimal. While incorporating the click factor did help, the gain was modest (0.02). Likely, this is because of the fact that a click through must occur after a read, and therefore the decision to click through is probably motivated less by the words in the title, and more by the content of the full story. While it is true that the content itself is dependent on the title, and, hence, there is some back propagation effect from the click through to the title, the fact still remains that the relationship between the tf-idf values of the title and the click through event is at best an indirect correlation.

Results

Due to the fact that we are dealing with skewed classes, resulting from the relatively few articles read by each user, we used the F1 score[2] as an indicator of accuracy. Our final results yielded an average accuracy of 0.46 per user per day. However, our performance did not improve as we increased the size of our training set; and while the average user accuracy fluctuated around the value of 0.45, the performance of the system diminished

for most users.

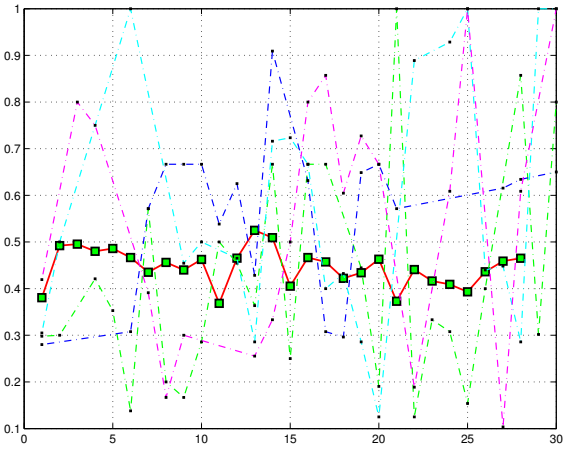

Figure 3: Accuracy as measured by daily F1 score for several users. The red solid line shows the average F1 score of all users and is hovering around 0.45 throughout our testing period

Our basic tf-idf contributed to 0.24 of the observed accuracy. Incorporating feed information boosted our accuracy by 0.14. Clustering yielded another 0.05 and incorporating click information increased our accuracy by 0.02

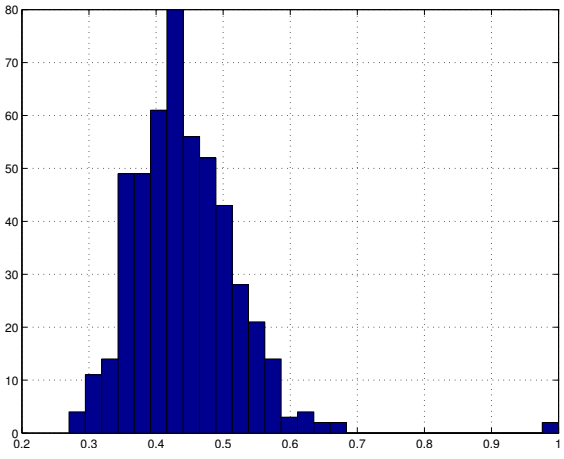

Figure 4: Histogram showing the distribution of average daily F1 scores across users

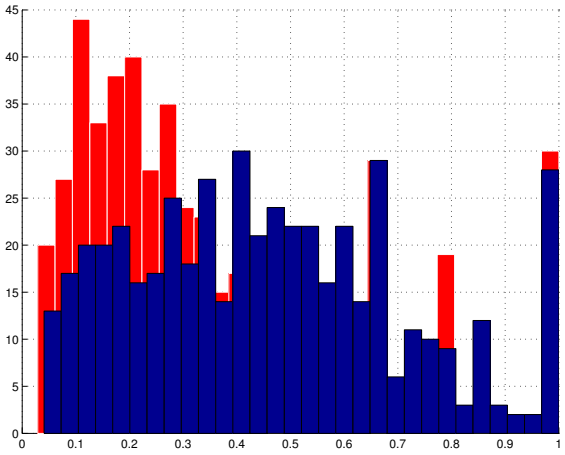

Figure 5: Histogram showing the distribution of daily F1 scores across users for the first day (in blue), and the last day (in red)

Conclusion and Future Work

We consider these results to be highly encouraging. We believe users will be satisfied with a system that could recommend articles with a 50% chance that they will opt to read them. However, we do believe we could have achieved significantly better results if were better able to filter out the set of stories a user was not exposed to. Furthermore, due to the limitations of current natural language processing technology to extract meaning from a single sentence, we believe better results could be achieved by training our algorithms on the entire text of an article, as opposed to its title alone. In addition, if we were to implement our system on a real live data set, our vocabulary would have to be dynamic, incorporating new terms that are introduced into the media. Such a system would also need to take into consideration the dimension of time by placing less weight on stories that appeared in the distant past, as opposed to stories that appeared in the recent past.

References

[1] Andrew Ng, CS 229, Class Lecture Topic: *Support Vector Machines*, NVIDIA Auditorium, Stanford University, October 21st, 2011

[2] Andrew Ng, CS 229, Class Lecture Topic: *Advice on Applying Machine Learning Algorithms*, NVIDIA Auditorium, Stanford University, November 7th, 2011

[3] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin., *LIBLINEAR: A Library for Large Linear Classification*, Journal of Machine Learning Research 9(2008), 1871-1874. Software available at http://www.csie.ntu.edu.tw/ cjlin/liblinear

[4] eMarketer, (2011, December 12), *Mobile Passes Print in Time-Spent Among US Adults* [Online]. Available: http://www.emarketer.com/ PressRelease.aspx?R=1008732

[5] Christopher D. Manning, Prabhakar Raghavan, Hinrich Schtze, *Introduction to Information Retrieval*. [Online] Available: http://informationretrieval.org

[6] Andrew Ng, CS 229, Class Lecture Topic: *Unsupervised Learning Algorithms*, NVIDIA Auditorium, Stanford University, November 28th, 2011

[7] Andrew Ng, CS 229, Class Lecture Topic: *Unsupervised Learning: Clustering*, NVIDIA Auditorium, Stanford University, November 7th, 2011