PREDICTING PREFERENCES
Analyzing Reading Behavior and News Preferences

Soravis Srinawakoon Potcharapol Suteparuk
Advised by Richard Socher

1 INTRODUCTION

News reading has gradually become a significant activity in our daily life as the world is saturated with new information. We consume so much information every day that it becomes extremely difficult to filter and extract only the interesting and relevant news to us. To improve readers' experience, we need to be able to accurately predict the new stories that are most probable for them to read. This reduces to predicting preferences which is a common problem of finding out which items are most relevant to each user and essentially rank or feed them to tailored users. In our study, we in particular look at the stories users click and read previously in a news-reading mobile application Pulse in order to predict which stories users are most likely to read in the next few days.

Similar challenge has been recently tackled by researchers in data mining and natural language processing because its implications can be applied to other popular areas such as search engine recommendation system. This paper considers applying different machine learning algorithms in the realm of supervised learning and unsupervised learning in attempt to predict the news stories that each user are most likely to read from a set of all available stories, which are much too large for the average users to parse and find the most relevant ones. In particular, using users' click and read history, we explore text categorization algorithms by comparing the accuracy of several supervised learning methods such as a simple Naive Bayes, L2-norm regularized logistic regression, and L1-norm regularized logistic regression with Lasso algorithm. We then move on to investigate common unsupervised learning technique such as k-mean clustering of users and a simple implementation of collaborative filtering.

2 DATASET AND TEXT REPRESENTATION

Given three set of input data from Pulse: 26,903 available stories at a given period, all stories read from a sample of 1,068 Pulse users, and all click-through stories from the sample sample of 1,068 Pulse users during the given period, we extract and represent all the available stories as feature vectors $X = \{x^{(i)}\}$ where $x^{(i)}$ corresponds to each story's input features. We represent this input feature using a simple form of TF-IDF term weight (term frequency times inverse document frequency) where $x^{(i)} \in \mathbb{R}^n$, $n = |W|$ is a total number of distinct terms appeared in all document. As a starting baseline, we represent each document using only its title and feed title so $|W|$ is total number of distinct words appeared in all title and feed title. This gives term j in document i a TF-IDF weight of

$$x_j^{(i)} = \begin{cases} 0 & \text{if A(i, j) = 0} \\ (1 + \ln A(i,j)) \ln(\frac{|S|}{A(j)}) & \text{otherwise,} \end{cases}$$

where $A(i,j)$ is the number of occurences of term j in a document i, $A(j)$ is the number of avalaible stories that contain the term j, and $|S|$ is total number of avalaible stories.

Later, we will explore a subset of our available data and represent each document by its title and its content. We will make a slight adjustment to our TF-IDF term weight by adding a cosine normalization to our feature vectors which has shown a significant improvement in prediction for large document size as it reduces the impact of document length [1].

Thus, the final normalized weight is:

$$x_j^{(i)} = \frac{non - normalized\ x_j^{(i)}}{\sqrt{\sum_{j'}^{n} x_{j'}^{(i)} \times x_{j'}^{(i)}}}$$

where we sum over all the terms j in the denominator but we denote it with j' to avoid confusion.

3 METHODOLOGY AND CLASSIFIER

In this part, we present all of the method and classifier used. The result of each will be present in part four where we evaluate each classifiers performance by measuring its RMSE, precision, recall and F-measure.

3.1 Features Selection

Our data set and in general the text categorization data are very sparse with unclear relationship between input features and class labels. Therefore, we try to select features that are most relevant using typical text preprocessing methods such as stop words removal, singletons removal, and stemming using Stanford JAVA NLP library [2]. Unless otherwise stated, we will treat our data set as the combined data set of users read stories and click-through stories.

3.2 Naive Bayes

Naive Bayes classification assumes that each feature in our data set is conditionally independent to each other. That is, for each feature $X^{(i)} = (x_1^{(i)}, x_2^{(i)}, \ldots, x_n^{(i)})$, which has a given label $y^{(i)} \in \{0, 1\}$, we can assume the following

$$P(X^{(i)} = X | y^{(i)} = r) = \prod_i P(x_j^{(i)} = x_j | y^{(i)} = r).$$

Apply this assumption to make our hypothesis as followed,

$$h_\theta(X) = argmax_r \frac{P(X|y = r)P(y = r)}{P(X)} = argmax_r \prod_i P(X = x_i | y = r)P(y = r).$$

In our model, we use the above formula to classify whether a specific user of the application is likely to read the given story (X), whose title includes words $x_1, x_2, x_3, \ldots$ etc. Based on the training data, we can calculate the conditional probability that each of these words occurs in the stories that the user read ($y = 1$) and did not read ($y = 0$). The product of these conditional probabilities times the prior probability gives each classification its likelihood for X. Our hypothesis outputs the class that has the higher corresponding likelihood.

3.3 Logistic Regression

The first baseline of our prediction is by using logistic regression to classify our data. This model is the special case of Generalized Linear Models (GLMs) for Bernoulli variable. The idea is to parameterize our hypothesis with the sigmoid function:

$$P(y|X, \theta) = (h_\theta(X))^y (1 - h_\theta(X))^y; \; h_\theta(X) = (1 + e^{-\theta^T X})^{-1}.$$

In our case, X represents the stories with title words as their features. $h_\theta(X)$ is our hypothesis and the probability of X being labeled 1 (i.e. read), and hence the equation above. We then can use the training data set to train our parameters θ by minimizing the negated log-likelihood

$$l(\theta) = -\sum_{i=1}^{n} \ln(1 + e^{-\theta^T x^{(i)} y^{(i)}})$$

which we will use later in part 3.4 and 3.5

3.4 L2-norm Regularized Logistic Regression

Taking our two data sets into account, we minimize a cost function that incorporates logistic regression for users read stories, logistic regression for users clickthrough stories, and a penalty L2-norm of the differences between θ_r and θ_c found by training logistic regression with read stories and clickthrough stories respectively.

Formally, we want to fit our parameters θ to classify whether the users read the stories and whether the users perform a clickthrough, and thus we want the two θ's to be a close approximation of each other. Thus we L2-regularize our dataset and θ can be found, using gradient descent, by minimizing

$$l(\theta) = l(\theta_r) + l(\theta_c) + \beta||\theta_r - \theta_c||$$

where β controls the relative weighting between trying to minimize the first two negated log-likelihood and of ensuring that the parameters found are as close to each other as possible. We chose β to be Mallows Cp as it has been demonstrated to be a good regularized parameter that addresses the issues of overfitting which is our main concern when developing our classifier. [3]

3.5 L1-norm Regularized Logistic Regression with Lasso Algorithm

Next, we L1-regularize our data set by using the well-known lasso algorithm which has shown to be powerful in solving text categorization problem [4]. We thus find θ_{lasso} by minimizing

$$l(\theta_{lasso}) = l(\theta) + \beta \sum_j |\theta_j|$$

where $\beta = Cp$ again controls the degree of regularization.

3.6 K-Mean Clustering

Now, we switch to unsupervised learning by using k-mean clustering. Specifically we group users and make predictions based on users being in the same cluster using cosine similarity. Now, the chosen k value is critical as the number of groups can significantly affect the accuracy result. We repeatedly tried various value of k until we settled with $k = 23$. This value can also be seen as a rough representation of number of topics or category in our news.

3.7 Collaborative Filtering

Widely used technique in suggesting news based on other users who have read similar articles. This simple method yet effective have been implemented extensively for example on Netflix for movie suggestion and Amazon for product recommendation. It indirectly takes into account a correlation between reading similar articles between users. Though simple, this approach capitalizes on a topic model by a nave assumption that if two users read something in common in the past, they should read something similar in the future.

4 RESULTS AND CONCLUSIONS

To compare our testing result, a simple accuracy percentage is useless because our data is very sparse so that a simple classifier that predicts zero all the time would achieve 99% accuracy. Instead, we measure our result by comparing its RMSE (root-mean-square-error), precision, recall and f-measure to get a better predictive indication [see table 1].

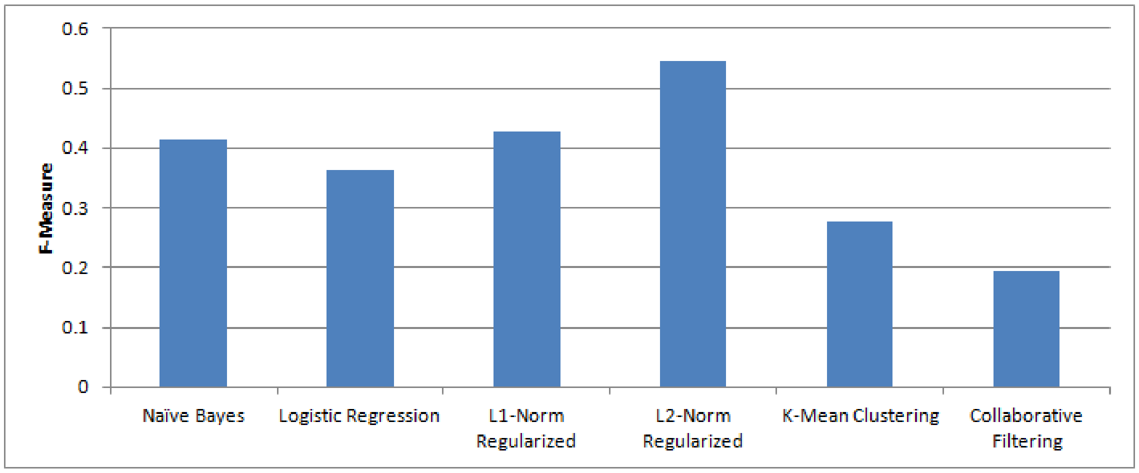
Algorithm	**RMSE**	**Precision**	**Recall**	**F-Measure**
Naive Bayes	0.78	0.4312	0.3974	0.4136
Logistic Regression	0.85	0.3892	0.3418	0.3640
L1-Norm Logistic Regression	0.80	0.4189	0.4375	0.4280
L2-Norm Logistic Regression	0.72	0.5725	0.5196	0.5448
K-Mean Clustering	0.94	0.2569	0.3041	0.2785
Collaborative Filtering	0.96	0.1201	0.4880	0.1928

Table 1: RMSE, precision, recall, and F-measure for different learning algorithm

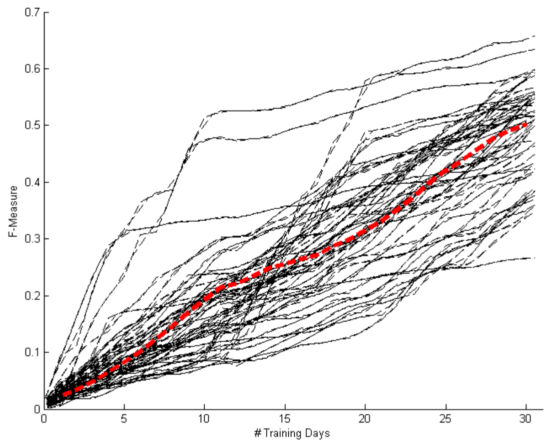
Surprisingly Naive Bayes performs better than our simple logistic regression. This is possibly because the way we incorporate the feed title into our title when we train and test our classifier as this has shown a slight improvement in our algorithms (since feed title does give a good indication of particular topic as some feed are really specific to one news category). Our logistic regression could not take into account the feed title as well because it merely tries to separate data in the feature space so it does not consider the distinction of each feed as well as Naive Bayes. L1-Norm LR performs a little bit better but again the model suffers from almost the exact same reason. L2-Norm however outperforms every algorithm and by far produces the best result. This makes sense because here we take into account the distinction between users read stories and click-through stories and it turns out that click-through stories provide a better indication of what users are likely to read in the future.

Unsupervised learning noticeably produces much worse result than supervised learning algorithms. Conceivably, k-mean clustering does not produce a good result because there is not really a direct relationship between TF-IDF and the group that it belongs to because the feature space we consider does not really model the topic model even if the cosine similarity implies that they belong to the same group. Similarly, our collaborative filtering uses TF-IDF to group similar users together but the TF-IDF does not provide a good insight about each topic model thus two users reading different topics that contain similar words will be grouped together. Also, because our algorithm only consider a pair of users each time, it is much harder to come up with a good match with only 1,000 users with more than 26,000 available stories. The number of users does not scale well with number of stories.

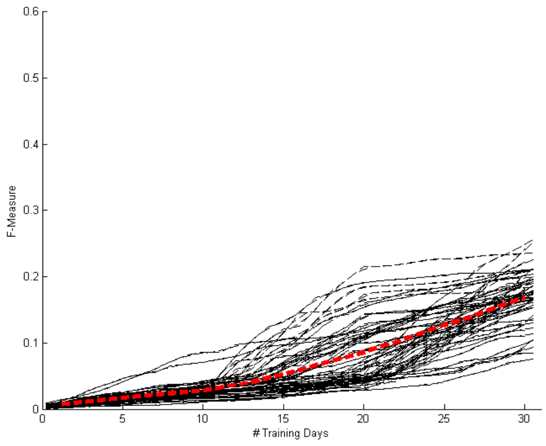
Note that it turns out when we probe through 30 stories (it already takes too long to crawl the website and extract the word so we can only use a small subset for testing) only produce a slightly better result than using stories title and feed as our news representation so we don't consider this approach here since 30 is too little to conclude anything.



F-measure compared between different learning algorithms



L2-Norm Regularized Logistic Regression



K-Mean Clustering

F-measure plot against the number of training day using L2-Norm regularized logistic regression [left] and K-Mean clustering [right]. Each black line represents each user for a sample of 50 users, while the red line shows the average F-measure. Some irregular patterns arise because some users read only at the certain period of a month so increasing the number of training days only changes f-measure slightly. For example, the graph on the left shows that some users read a lot of stories in the beginning (a sharp rise in f-measure which allows us to accurately predict his/her top stories later on. These irrigular patterns also show that our classifier's predictive ability also relies on the structure of the stories each user read because if the users read randomly, clearly the classifier will perform poorly whereas users who read a lot of repetitive stories will allow our classifier to perform well. We present here the contrast between two algorithms for comparison and clearly L2-Norm regularized logistic regression performs much better than k-mean clustering. Again, this is perhaps due to the underlying structure of our feature space that does not correlate well with clustering algorithm.

5 FUTURE WORKS

Further application of predicting news preferences can be far-reaching and significant as this essentially reduces to a problem of finding preferences and recommending particular group of similar items based on past input. Based on our observation thus far, the model we use did not achieve a considerably practical result perhaps because it does not directly link words (features) to its actual topic so the next reasonable step to take is to consider a model that takes topic and word clustering into account. Latent Dirichlet Allocation which gives a topic model that cluster words with similar things can be explored. Additionally, we did not have enough time to incorporate automatic relevance determination [5] and named-entity recognition which classifies words with similar spelling but different semantic differently based on the context. These pre-processing on the feature space should relate our feature space closer to its actual topic model. Based on our result, it can be seen that some of the output articles contain similar words but these words can have different meaning so this system can improve our classifier as the best existing NER system (MUC-7) has shown 92% recall and 93% precision with errors lie mostly in the entries that lack obvious English rules or condition of context [6]. We should also look into how to incorporate our click-through data into our classifier model so that it generates a more meaningful data rather than something similar to read stories as it has been shown to produce good result with text classification [7].

6 REFERENCES

1. Salton, G. and Buckley, C. (1988), "Term-Weighting Approaches in Automatic Text Retrieval, *Information Processing and Management*, 24, 513-519.

2. *Stanford corenlp tools version 1.2.0.* (2011, Sep 14). Retrieved from http://nlp.stanford.edu/software/corenlp.shtml.

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

4. Tibshirani, R.: Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society, Series B (Methodological) 58(1), 267288 (1996).

5. Liitiainen, E. (2006). Automatic relevance determination. In Finland: Retrieved from http://www.cis.hut.fi/Opinnot/T-61.6040/presentations_s06/presentation_elia.pdf.

6. Black W., Rinaldi F., and Mowatt D., "Facile: Description of the NE System Used for MUC-7, *Department of Language Engineering UMIS*: Retrieved from http://www-nlpir.nist.gov/related_projects/muc/proceedings/muc_7_proceedings/facile_muc7.pdf.

7. Joachims, T.: Optimizing search engines using clickthrough data. In: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), pp. 133142 (2002).