Personalized News Prediction and Recommendation

Abhishek Arora
arorabhi@stanford.edu
Dept. of Electrical Engineering
Stanford University

Preyas Shah
preyas@stanford.edu
Dept. of Mechanical Engineering
Stanford University

Abstract: There exist many web based news provider applications (e.g. Pulse News reader application for iPhone/iPad and Android platforms) that gather news articles from the myriad 'feeds' the user is subscribed to and displays them with-in each feed category. A feed refers to a stream of news articles that may represent a news category (e.g. Gadgets, Technology, Sports, etc.) or may be very general (e.g. RSS feed provided by some newspaper for top articles, etc.). In this project, we analyze and develop machine learning techniques for *personalized* feed based news prediction and recommendation.

Keywords: Natural Language Processing, Information Retrieval, Machine Learning.

1.1 Introduction:

Of all the things people read these days, news has a large effect on their opinion formation regarding different issues. Also, newspapers and news services may manipulate the general mood of the people by showing them more sad news or happy news. Apart from opinion formation, one would also be interested in showing the reader the kind of news she is most interested in. Unlike conventional newspapers (which are same for all), the internet has revolutionized the information flow and access in user specific manner. Now, there exist many news services like Pulse application for iPhone/iPad that can provide personalized user experience by giving the users freedom to subscribe to the feeds (news sources) they are most interested in. Most of these services have the same set of news articles for each user in a particular feed.

Thus, one level of personalization is where each user sees articles from a certain set of feeds that she can subscribe to (Figure 1), while articles in each feed remain same for all users. We propose a further level of personalization in which news articles in each feed are also personalized according to the user's interest.

In order to accomplish this, we apply natural language processing and information retrieval techniques to extract features about the news articles and use these features in a machine learning setting to learn user behavior and interests.

2 Data

The dataset comes from a local startup, Alphonse Labs, the developers of the Pulse application. The data consists of the activities of 1068 users for the month of August 2011. We have three types of files available:

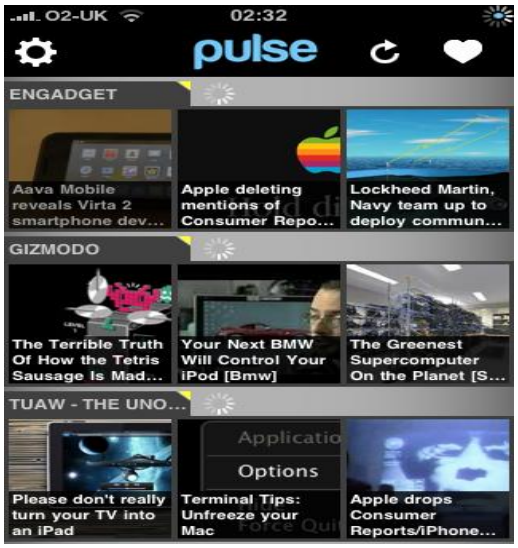

Figure 1: A sample snapshot of pulse application.

1) **Stories**: This consists of all the news articles (stories) available for reading with the following information:
article-url, article title, feed-url, feed title, timestamp of first read (a proxy for when the article appeared in the feed). There were circa 1,65,000 unique stories, ~450 unique feeds. A story is identified by the *(article-url, feed-url)* tuple.

2) **UserActivity**: This consists of the stories the users have read, with the following information:
user id, story url, story title, feed url, feed title, timestamp when user reads

3) **ClickThroughs**: This consists of the stories that users read in the web mode. Usually, users see stories in text mode in the app, a click-through event is defined as clicking a link to view a story in web mode that may indicate a higher user interest for that story. We have the following format for each entry in this file:
user id, story url, story title, feed url, feed title, timestamp.

2.1 Curing the data:

The first version of the dataset provided didn't have the timestamp information for 'Stories' file. Thus, we had no information about when that story became available. This

information is required to determine which stories the user might have seen (but not read). This is important for constructing the class labels (0-read, 1-unread). On our request, the time stamp information was incorporated.

We also faced problem of ambiguous results and realized after four weeks that the data provided to us had malformed feed-urls. Significant number of feed-urls in the file-'UserActivity' did not match the corresponding feed-urls in the file 'Stories'. We again had to request for the corrected data.

Also, lot of entries in the files had malformed format (e.g. different number of columns, bad time stamp, a lot of duplicate entries just differing by timestamp for a given user on a given day). So, we had tough time dealing with the data.

2.2 Data Organization

We hashed feed-urls and article-urls using md5 algorithm to reduce memory requirements and for fast access. The data was then organized as:

- New stories arriving on each of the 31 days.
- Stories read by each user on each of the days.
- Stories each user saw (read/not read) for each day. We assume if a user views a story from a feed, he has looked at all the stories from that feed. Also, a user does not necessarily read stories from all the feeds she is subscribed to.

3 Feature Extraction

We characterize a user's reading habit/behavior as the one that has two components to it:

1. The user specific component that captures the sole user's intrinsic behavior
2. The influence/ trend of the other people who have similar interests as the user. This captures how much the user is influenced by the trends in his community.

The first component is estimated by extracting the features from the articles the user reads (ref. Section **3.1**). The second component is estimated by clustering all the users to apply collaborative filtering techniques (ref Section **3.2**). We then add these two components to the features of each article.

Note that not all articles in a feed are new arrivals. All the features for a given story that is available for reading in a feed (today) are constructed by looking at only the history of what has happened till today.

3.1 Text feature extraction

As we can see from Figure 1, user first sees the title of the article. Thus, we extract the text-based features from the title of the article. There are many standard techniques available for text feature extraction. Some examples are:

3.1.1 TF-IDF (Term Frequency-Inverse Document Frequency) vector formulation: TF-IDF score of a word for a document tells us the importance of the word for that document relative to the other documents in the collection. In our case, document refers to the article title.

3.1.2 LSI (Latent Semantic Indexing): LSI uses SVD to identify the relationship between the terms and concepts contained in an unstructured collection of text. LSI essentially tries to identify the similarity between the words based on the context. It is similar to PCA in the sense that it projects the features into a lower dimensional space yet improves information retrieval.

3.1.3 LDA (Latent Dirichlet Allocation): LDA is an example of topic modeling. It assumes that the set of documents belong to a small set of topics and the words appearing in the documents are generated on the basis of the underlying topics. We can specify the number of topics. A new document is represented by the probabilities of each topic occurring in the document.

We have used the above three techniques to extract the features from the title text. Also, we preprocess the text by removing stop words, considered stemming the words**,** removed less relevant words/parts of speech by tagging the article title (using POS taggers like the Stanford POS Tagger or a NLTK Penn Treebank Tagger), replaced catchy symbols like '?', three dots '...' (that invoke curiosity among the readers) with fixed labels.

3.2 The view of an article from other user's perspective

In this section, we try to estimate how likely a general user would read a given article.

We also try to estimate how likely is the user to read the article, given the information of how likely other similar users are to read the same article.

3.2.1 General Article Popularity

A given article has some prior probability of how likely a random user may read it. We estimate this probability as:

$$p = \frac{number\ of\ users\ who\ read\ the\ article}{number\ of\ users\ who\ have\ seen\ the\ article} \quad (1)$$

Thus, when constructing this feature for an article for a given user, we see how many people have seen and read this article until the present day. Since, not all articles in a feed are new arrivals, an older article may give us much better estimate of this probability.

The question is how to detect whether a user has seen the article but not read it? We assume that all the articles in a given feed has been seen by an user if he/she has read at least one article from that feed on the day for which we are constructing the features.

3.2.2 Collaborative filtering

We do three types of clustering using *K-means*. We cluster the users based on the (1) number of times the user has opened each feed till the present day, (2) the number of articles from each feed the user has read till the present day and (3) kind of

feeds the user is subscribed to till the present day. Figure 2 shows the number of users in each cluster for the three techniques (cluster indices for the three techniques have no correspondence). With reference to Fig. **2**, we can see that techniques 1 and 2 show single high peak => relatively large number of users in one cluster, while technique (3) shows a roughly equal distribution of users across the clusters. The concentrated high peaks for techniques 1 and 2 may be due to the fact that some feeds are very viral, a lot of users are subscribed to them, and they fetch most user activity. The clusters having the peaks may be attributable to these viral feeds.

The score for a given article-***a*** for a given user-***u*** is calculated in the following two ways:

$$S_{1(u,a)} = \frac{\sum_{i=1}^{|c_u|} I\{i_a==1\}\varphi(u,i)}{\sum_{i=1}^{|c_u|} \varphi(u,i)} \quad (2)$$

$$S_{2(u,a)} = \frac{\sum_{i=1}^{|c_u|} I\{i_a==1\}}{|c_u|} \quad (3)$$

where c_u represents the cluster set of user-***u*** , $\varphi(u,i)$ is the cosine similarity between user-***u*** and user-***i,*** and $I\{i_a == 1\}$is the indicator function of whether user-***i*** has read the article-***a.*** We calculate these scores for all the three clustering techniques.

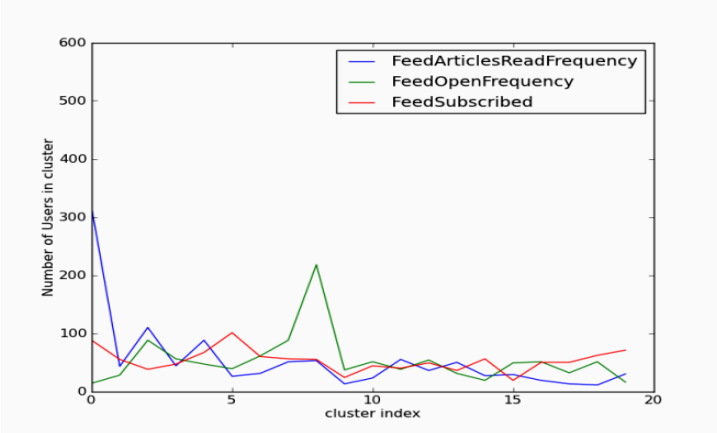

Figure 2: Number of users in each cluster (K=20 in K-means) for the three clustering techniques on day-15.

4. Supervised Learning model

Certainly, this is a binary classification problem where for a given user, class-0 represents that the user has not read the article (but has seen it on the feed), while class-1 represents that the user has read the article. We can in fact define a third class, class-2 indicating that the user has not seen the article yet (that is, she has not read any article from the feed but has subscribed to it). We just keep two classes for simplicity. It also makes sense to predict which articles a user reads given a feed rather than also predicting which feed a user will read from- for there is no fixed behavior behind not choosing to read from a particular feed for a given day. It also depends on in what order the feeds appear in different pages of the app.

We form feature vectors for each article using techniques in Section 3, and use these feature vectors in a logistic regression classifier.

4.1 Metric for performance evaluation

Note that the problem is highly unbalanced. We have lot more 0s than 1s as a user normally reads around 5-25% of the articles of what she sees in the feeds. Thus, we cannot use accuracy as our measure for even a default 0 prediction has a good accuracy. Therefore, we are interested in precision and recall. For recommendation of articles, we would like to recommend the user ***k*** articles and would want that we have good precision and recall for this set of ***k*** articles. Thus, we can plot precision-recall curves and calculate area under the precision recall curve. A higher area implies better performance. Also, we can use F1 score as metric for performance evaluation as it captures both precision and recall by giving equal weights to them. We can also use F2 score if we want to give more weight to recall than precision.

Precision-Recall curves: For plotting precision-recall curves, we sort the testing samples with the decreasing order of the class-1 classification probability. We then pick top ***x*** samples, and calculate precision and recall in this sample set. As we vary ***x***, we get different precision-recall values on the plot.

4.2 Optimal Model Parameters

Since the problem is highly unbalanced, if we simply use classification model with equal weights to classes 1 and 0, we would obtain almost all 0s as the predicted output since the classifier optimizes the accuracy by default in most of the packages available. In order to account for the unbalanced nature of the problem, we give more weight to class-1 than class-0. We penalize the misclassification of class-1 more heavily than misclassification of class-0. But we don't know what weight (***w)*** we must give to class-1. Also, we must choose optimal regularization parameter, **C**, with the f-score as objective. To find optimal ***w*** and C, we do grid search in the grid:

w x **C** = [10,30,50,70,90,110,130,150] x [10^{-3}, 10^{-2}, 10^{-1},1,10]

We choose the combination that maximizes the required metric (section 4.2) by doing 3-fold cross validation on the training data.

4.3 Training and Testing Methodology

We fit the classifier for each user. For training till day-n, we start from day-1 and proceed day by day by including the samples on each day until we reach day-n. In the process, for each day, we consider only the feeds from which the user has read at least one article. Articles read have label-1 while other articles in the considered feed have label-0. We don't just include articles from all feeds the user is subscribed to, because those articles were not seen for more reasons that just one: not enough time to read through all feeds, not wanting to read anything else on 'that' day, only more interested in page1 feeds but not uninterested in page2 feeds, etc. So, not reading any article from a feed doesn't necessarily mean that all the

articles in that feed don't capture the user's interest. Hence, we don't include these feeds, for accounting them lead to having labels-0 for the articles in these feed and we would be counting these articles as un-interesting articles in the training, also making the problem more unbalanced.

For testing, we just test on n+1 to n+5 days (i.e. next 5 days). We choose next five days for testing because for recommendation purpose, we would be interested in recommending articles in the near future (within 5 days) and user's interest may change over time. So, a 5-day window seemed reasonable. While testing, we again just include those feeds from which the user has read at least one article. This is because we are building feed based recommendation system. For a given feed, our model will recommend articles in that feed. *Hence, if we do well in predicting what articles user will read **given** a feed, we are doing a good job.* Other option is to include all the feeds the user is subscribed to, for testing. This is also a possible new direction, as now we will also need to consider feed preferences for the user while building feature vector and then predict what feed and what article in it the user will read.

5. Results

We plot the precision-recall curve averaged over 20 random users for whom we have enough training data. Most of the users don't use pulse app. daily and hence for them we have little data and it makes little sense in predicting anything for them, though we can still recommend and effectively influence their reading habits!

5.1 Feature Vector with just text features

In this section, we consider constructing the feature vector by only using text based features from the article titles.

Firstly, we show results by just including TF-IDF/TF as the feature vector. We see that the basic TF-IDF-based model performs better than the one with TF-based. With different types of text processing like stop-words removal and stemming, we plot the performance curves for the TF-IDF based model. We saw that (ref. Fig. **3**) there wasn't a significant effect of stemming and stop word removal.

We then compared the performance of the TF and TF-IDF based feature vector model (with model parameters ,class-1 weight(w) and regularization parameter(C), optimized for F1 and F2 scores) (ref. Fig. **4**). We saw that the curves are better for TF-IDF based model for both F1 and F2 as objectives. We also see that the curves for F1 and F2 have the same behavior for a given model, so we optimized C and *w* using F1 score in the plots thenceforth.

We then generated the precision-recall curves for the case where we implement LSI and LDA on the document TF-IDF features (ref Fig. **5**,**6**,**7**).

We saw that for number topics in LDA being 80, we got better performance (ref. Fig. **5**)

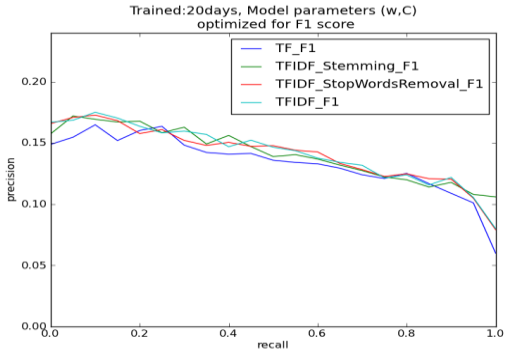

Figure 3: Precision-Recall Curves when feature vector was composed of TF/TF-IDF only. Different text processing tested.

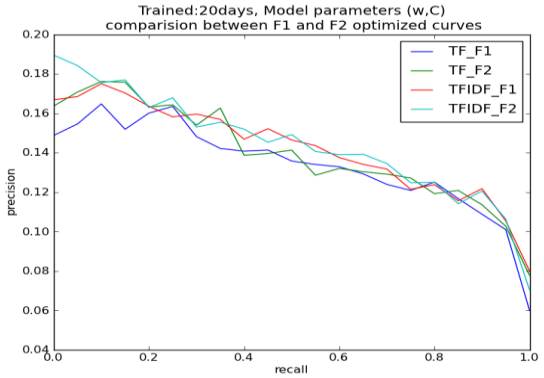

Figure 4: Precision-Recall curves when feature vector was composed of TF-IDF / TF.

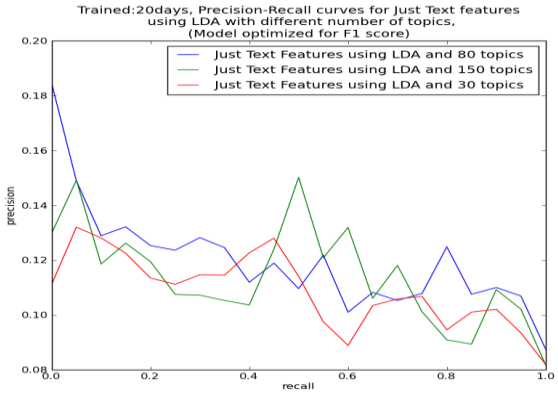

Figure 5: Precision-Recall curves when LDA was implemented.

Figure 6: Precision-Recall curves when LSI was implemented.

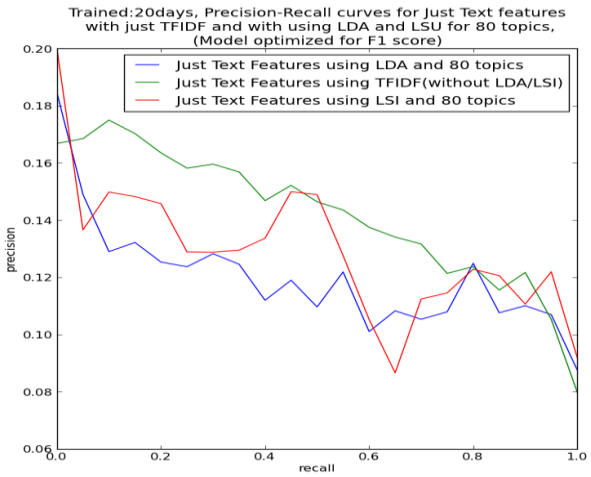

Figure 7: Precision-Recall curves comparison for LDA, LSI and just TF-IDF without any feature reduction.

We also saw that it did not make much difference by varying the number of topics while implementing LSI (ref. Fig. **6**).

Also, from Fig. **7**, we can see that purely TF-IDF based model performed slightly better than LSI/LDA (LSI and LDA applied on TF-IDF feature vector) model. This may be attributed to the fact that despite topic modeling, there is probably not enough topical content in article headlines and more of proper nouns or NE (named-entity) content. Also, the grid search based optimization brings up the performance of purely TF-IDF based model to the same level of the other two. We also saw that grid search did not affect the performance of the other two models much, which supports the above hypothesis.

Since, just TF-IDF based feature vector does better than the LDA/LSI models, we use just TF-IDF based features for the discussion henceforth.

5.2 Collaborative Filtering based feature vectors

We then tried clustering based on the three different techniques as mentioned in section 3.2.2. For each technique, we calculated the metrics $S_{\mathbf{1}(u,a)}$ (metric1) and $S_{\mathbf{2}(u,a)}$ (metric2) and added these metrics as features to the basic TFIDF based feature vector (i.e. we added 3 (techniques) x 2 (metrics) = 6 more features) and plotted the performance curves for different number of clusters (ref. Fig. **8**). As we can see, clustering significantly improved the performance! Moreover, number of clusters = 7 seems most appropriate.

Now, we try to evaluate which of the six clustering based features contributes the most to the improvement. So, instead of adding all the six features at a time, we add these features (metrics for each technique) separately as an additional feature to the basic TFIDF based feature vector and plot the performance curves (ref. Fig. **9**). The technique 1 (FeedReadFreq based clustering) contributes the most to the improvement followed by technique 2 (FeedOpenFreq based) and technique 3 (FeedSubscribed based). Also note that both metrics (S_1 and S_2) have similar effect for each clustering technique.

Since the best performance was seen for 7 clusters, use 7 clusters and add the six features to the basic TFIDF based feature vector henceforth.

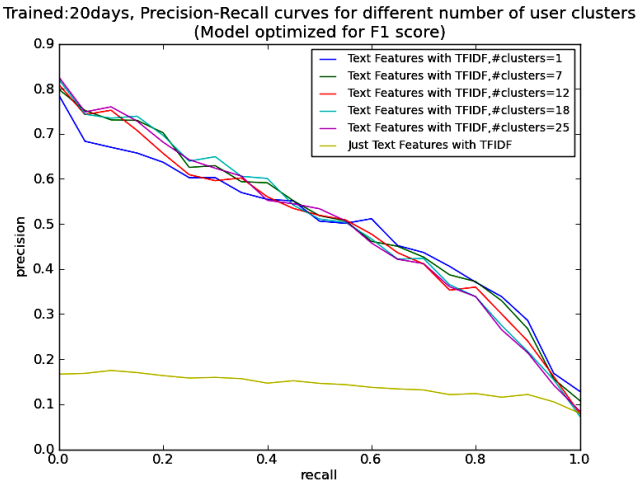

Figure 8: Precision-Recall curves for different number of clusters

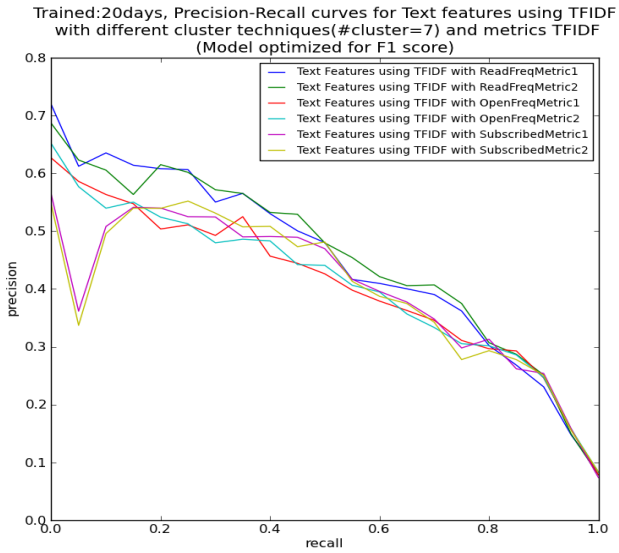

Figure 9: Precision-Recall curves for different metrics and different clustering models

We now check the effect of general article popularity (section 3.2.1) when added as a feature to the TF-IDF based features. Note that we already have a plot for just a single cluster in Fig. **8**, which captures general article popularity already. Thus, the general article popularity score doesn't make significant contribution.

5.3 Putting everything together

We finally plotted precision-recall curves with 7 user clusters, including the six clustering based features(two metrics for each of the three clustering techniques) with TF-IDF based text features for different training durations: 1-10, 1-15, 1-20 and 1-25 days. We test on next five days for each training duration. These are average curves for 30 random users. Since, the purely TF-IDF-based model is only slightly better than

LSI/LDA based models, we did not plot the same for the latter 2 cases (ref. Fig. **10**).

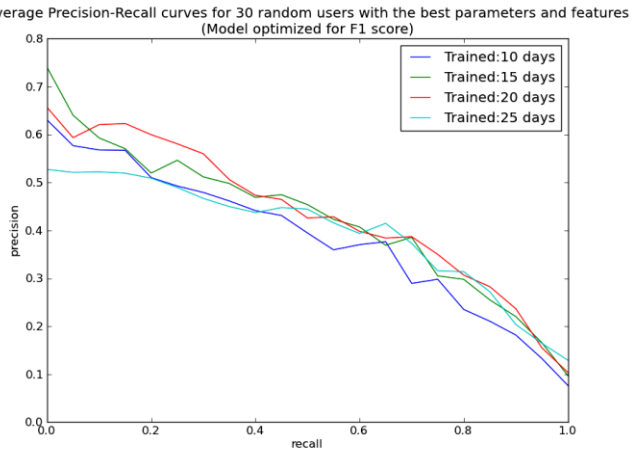

Figure 10: Precision-Recall curves for 30 random users with best parameters and features

Surprisingly, we don't see much effect of the number of training days. We expect that as we include more training examples, we will get improvement. This may be due to the fact that the news habits of people change over time and hence adding the past data is not very useful, rather could degrade the performance as we can see (lower curve for training till 25 day, training till 20 day seems most appropriate).

5.4. Extension

In the above analysis, we have not really used feed specific information while constructing the feature vector (other than in clustering techniques). One can think that a user has feed specific article selection behavior. Hence, one think of feed specific classifiers for each user. Since, it would be slightly cumbersome to maintain a different classifier for each feed and for each user, we can alternatively add feed specific features. For example, we can add feed reading probability for the user for each feed the user is subscribed to, and to which feed the concerned article belongs to, as features. (note that feed is a categorical variable). We tried going in this direction, but the results were not exciting.

6. Conclusion:

We have successfully applied natural language processing, information retrieval and machine learning techniques in developing a feed based personalized news prediction/ recommendation system. We observed that TF-IDF based model performs slightly better than LSI/LDA models in the present problem setting with the given data. As user clustering shows significant improvement in the performance, it can be said that similar users have a common view about a given article. Also, the complete user pool can be roughly divided into seven to ten different communities representing news genres/categories.

7. Acknowledgements:

We would like to acknowledge Richard Socher (Head TA, CS229, Stanford Univ, Fall 2011) for giving us initial directions and guiding us through the project.

8. References:

[1] Scikit: http://scikit-learn.org/stable/

[2] Gensim: http://radimrehurek.com/gensim/

[3] Wikipedia http://www.wikipedia.com/

[4] Evaluation of Text Clustering Methods Ysing WordNet. Abdelmalek Amine, Zakaria Elberrichi, Michel Simonet. The International Arab Journal of Information Technology, Vol.7, No.4. October 2010. http://www.ccis2k.org/iajit/PDF/vol.7,no.4/1097final.pdf

[5] Probabilistic Topic Models. Mark Stevens, University of California, Irvine. Tom Griffiths, Brown University. http://www.cs.umass.edu/~wallach/courses/cs791ss/readings/steyvers06probabilistic.pdf (LDA)

[6] Google News Personalization: Scalable Online Collaborative Filtering. Abhinandan Da, Mayur Datar, Ashutosh Garg. WWW 2007/Track: Industrial Practice and Experience. Ay 8-12, 2007. Banff. Alberta, Canada. http://www2007.org/papers/paper570.pdf