News Feed Optimizer

Final Report

Brennan Burns
Vijay Harid
Fadi Zoghzoghy

1. INTRODUCTION

Each day, millions of people across the globe read thousands of news stories regarding many of the different, ever changing realities of the world. As such, the news industry has become a business like any other - one of maximizing profits by determining and delivering the most useful or desirable product to the customer. A major part of this is the proper advertising of news stories to the end-user – the more refined and personalized the advertising, the more effective.

It is for this reason that the entire idea of a news feed was created. By supplying the customer, or user, with a constant stream of not only current news topics, but those specifically of interest to that individual user, the likelihood that the user will read an article on the news company's site increases. Obviously, in order to accomplish this goal, there needs to be an efficient automated way to determine what articles are most likely to interest a user. Enter machine learning.

2. PRE-PROCESSING

As with all machine learning problems, the biggest hurdle is acquiring a good data set. This is more than simply having a sufficiently large data set for the different methodologies and techniques to work well, but also encompasses the problem of having well posed data. In an ideal world, all data would be recorded and stored in such a form, but this is obviously not the case, and as such, quite a bit of pre-processing was required to massage the data into a working form.

The first step of the pre-processing was to parse and reformat the original data. Part of this process included the recognition and removal of duplicate entries, which were rampant throughout the data.

The next step was to build a dictionary of all the words in the data set. For this dictionary, all of the words were forced into lower case, all symbols and punctuation were removed, and the resultant words were stored in a single large vector of approximately 93,000 words. With this vector in hand, the TF-IDFs for each user for each day were calculated, and used in the initial implementation, discussed below. This giant feature set, however, led to the over-fitting of the data, requiring the number of features to be reduced, as will be discussed in *Section 3*.

3. INITIAL APPROACH

With all of the TF-IDF vectors created, logistic regression was applied to each user for each day, allowing for the basic prediction of which articles the user will read. While this extremely simple approach results in a seemingly excellent accuracy of ~98%, it is a meaningless result, which is obtained by always predicting that a given user will not read a particular article. This comes from the fact that a given user will only be exposed to, let alone read, an extremely small number of articles relative to the total amount. In order to accurately predict what a given user would read in a meaningful way, the articles considered for that user need to be reduced to those which he is likely to have been exposed to. Since all of the articles in the data set come from a specified feed URL, it is reasonable to assume that a given user was only exposed to articles from the same feeds from which he accessed the

articles he read.

The first step in this reduction was to separate the articles by their respective feed URLs, of which there were approximately 450. Once the feeds were segregated, logistic regression could be applied to only the articles associated with the feeds a given user accessed, which was, on average, between four to seven feeds per user. While this greatly reduced the total number of articles considered, the number of articles in those handful of feeds a user accessed could still be quite a bit larger than the user could have reasonably been exposed to.

In order to balance the negative and positive instances in the training and testing process, it was assumed that for every article a user read, there was a second article that he was exposed to, from the same feed, that he did not read. This second, non-read, article was randomly selected from the associated feed. This balancing prevents the user from being punished for not reading articles which he was not exposed to.

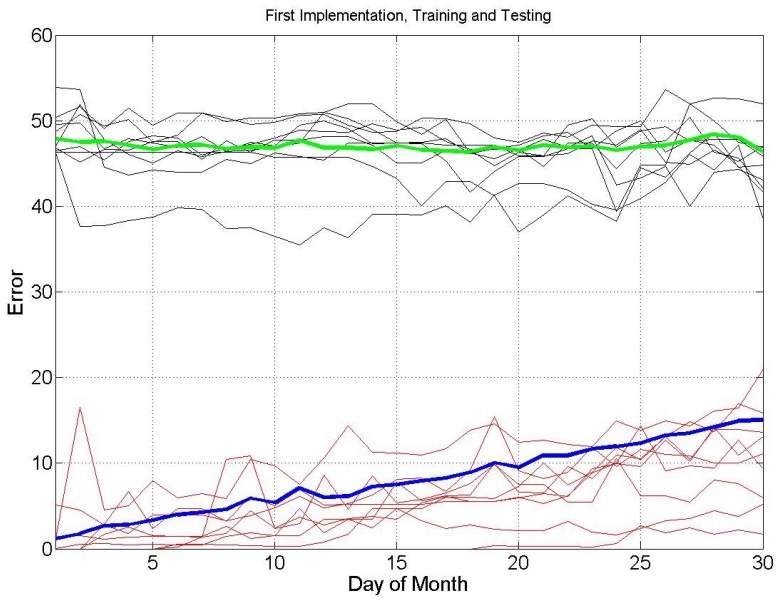

Figure 1: Training and Testing Error of Logistic Regression – In green is the mean testing error, which should be decreasing over time, and the mean training error is in blue. The red and black curves are the respective training and testing error for eight individual users.

With the number of articles sufficiently reduced, both logistic regression and SVM, via Liblinear, were applied to predict which articles a given user would read. While the hurdle of the false 98% accuracy had been overcome, it became immediately apparent that data was being over-fit by these algorithms. *Figure 1* shows the training and testing error from the application of logistic regression.

The application of SVM resulted in very similar behavior, meaning that the total number of features, or words in the dictionary, needed to be reduced, as discussed previously in *Section 2*. From the initial ~93,000 words, the dictionary was reduced to approximately 6,000 words by eliminating stop words, such as articles, and those that rarely occurred throughout the month. While this reduction greatly increased the computational efficiency of the algorithm, the shear amount of data that needed to be process was so large that most PCs were unable to run the computations. These computations were only successfully run when using a 16 core computer cluster, which still took quite a bit of time to process everything, making it difficult to debug and develop. Furthermore, this feature reduction did not cure the over-fitting problem, which required a more intricate approach.

4. CLUSTERING APPROACH

The problem of over-fitting, mentioned in the previous section, is caused by having too little data per user. Since the data set only covers the reading habits of the users throughout the month of August, the total number of articles a given user read is relatively small. By clustering users based upon their reading preferences, this problem can be avoided, since the total data volume per cluster is significantly larger than the volume per individual users. Another major advantage of clusters is that once a user has been identified as belonging to a particular cluster, articles can be recommended to that user based upon what other users have read. For example, if user *i* has been identified as being in the same cluster as user *j*, articles that user *j* has read will be recommended to user *i*, since both users are from the same cluster and are likely to be interested in the same articles.

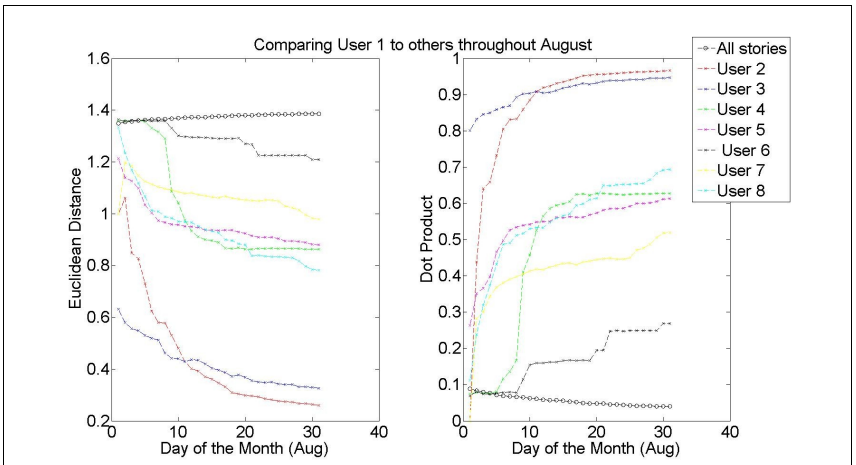

Figure 2: User comparison relative to user 1 throughout August. The figure on the left shows the distance between the TF-IDF vectors of user 1 *and seven other users (*users 2-8*) and* all-stories*, which is the base case in which a user reads all the stories every day, throughout the month of August. The figure on the right is similar, but computes the dot product instead.*

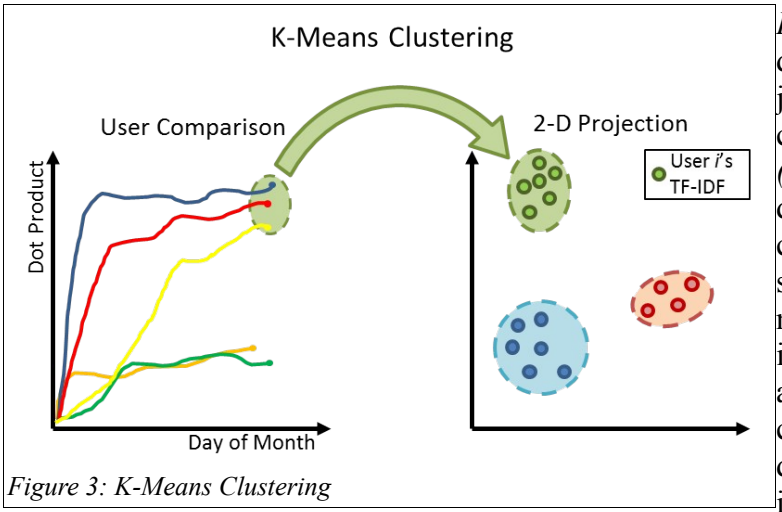

Figure 3: K-Means Clustering

Figure 2 shows a comparison of several different users and how they can be justifiably clustered. For example, *user 1* can be clustered together with *user 2 (blue)* and *user 3 (red)* as the Euclidean distance between their corresponding cumulative TF-IDF vectors is relatively small, and the dot products of the normalized TF-IDF vectors are large. This indicates the presence of strong similarities among these users, allowing them to be clustered via K-Means clustering by cosine distances of the TF-IDF vectors, as shown in *Figure 3*.

These clusters were found by taking the TF-IDF vectors of 350 users on day 31, and finding the cosine distances between these users, using them as parameters for k-mean clustering into eight clusters, which was found to be a good compromise between accurate clusters and computational efficiency. These eight clusters, shown in *Figure 4*, were then further optimized with 150 additional users by identifying their most appropriate cluster based on the cosine distance between their TF-IDF vector, and the average TF-IDF vectors of the clusters. Each of the remaining 500 users are then identified as being part of a cluster every day, using the same method. It was found that most users did not change

clusters after the fifth day they read stories, meaning that the stories they read could be accurately predicted.

Once the users were clustered, several different approaches could be used to actually predict whether a specific article would be read by a given user. As shown in *Figure 5*, a given article is tested by user *i*'s cluster, be it by majority voting or treating the cluster as a super-user, as discussed below. Depending on the resultant decision by the cluster regarding the given article, it will be recommended to the user in question.

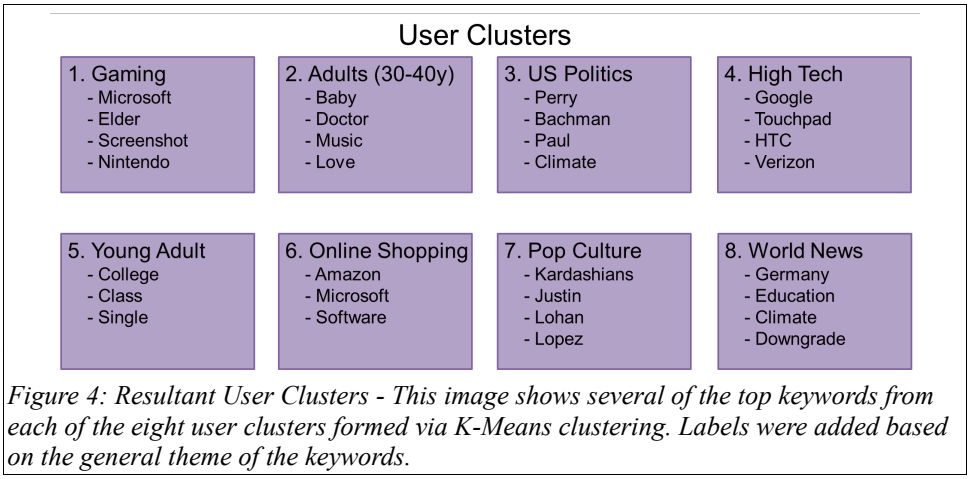

Figure 4: Resultant User Clusters - This image shows several of the top keywords from each of the eight user clusters formed via K-Means clustering. Labels were added based on the general theme of the keywords.

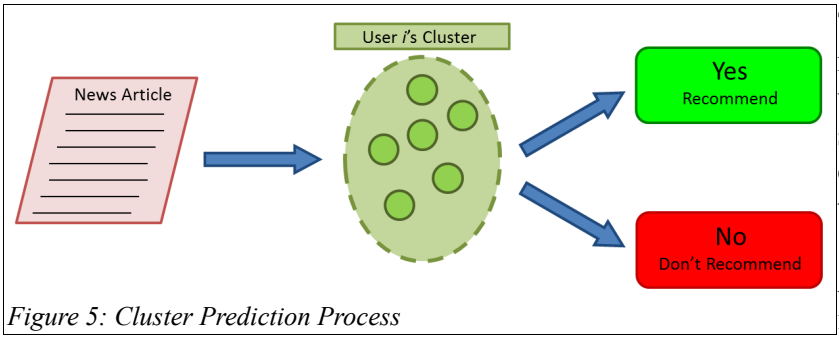

Figure 5: Cluster Prediction Process

The first of the two decision methodologies used, that of majority voting, was implemented by using a system of weak, uncorrelated classifications. The main idea being that for the given article was tested via SVM for every user within the cluster. All of the resultant decisions for each individual user were tallied, and whichever decision had a simple majority of votes, won, and was passed on as the recommendation for the article.

The second decision methodology was to treat each cluster as a super-user of sorts, meaning that all of the stories and all of the feeds that the users within a given cluster were treated as if they were accessed by one giant "super-user." In this case, the decision regarding the given article was based on the application of SVM to the entire group of articles related to this super-user, and was then used as the recommendation for the article.

5. RESULTS

The application of majority voting via the clusters turned out to be less than stellar. The resulting f1 scores, shown in *Figure 6*, averaged approximately 0.5 over 30 days for each of the eight clusters. A large part of this is due to the fact that SVM is being applied for each article to each user within the appropriate cluster. These votes are then tallied to determine the proper recommendation. Since SVM is still being applied on an individual basis, the expected accuracy is on the same order as individual prediction – and therefore still random, but is averaged over a large number of users.

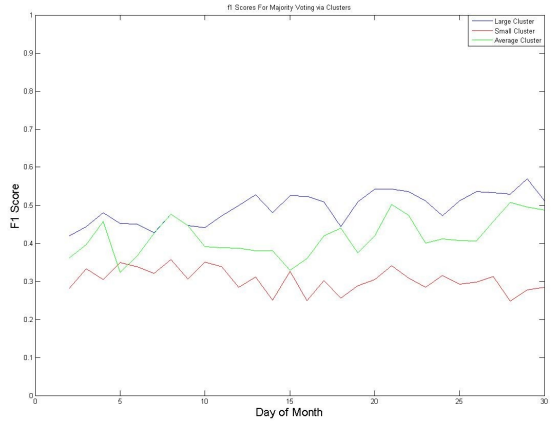

Figure 6: f1 Scores of Clusters with Majority Voting - This plot shows several f1 scores of clusters used in majority voting over the month of August. The blue curve is of a large cluster, the red of a small cluster, and the green is the mean f1 score of all clusters. As can be seen, larger clusters tend to have better f1 scores, which increase over the course of the month.

As expected, the application of a super-user via the clusters turned out to have slightly better accuracy than both majority voting and individual predictions, as demonstrated by the f1 scores of the clusters, shown in *Figure* 7. This is largely due to the fact that the super user approach has a far larger quantity of data than what is available to individual users. This indicates that a far larger data set is desirable to obtain more accurate results.

As discussed in *section 2*, the vast quantity of data required for the calculations ran on a 16-core cluster, often for many hours at a time. These time scales limited the amount of optimization that could feasibly be done. One such future optimization would be to change the base time scale from days to hours, which would improve the accuracy of these predictions – since articles read in the morning by one user could be recommended to a different user in the same cluster later in the day. It would, however, increase the computational requirements of the predictions.

6. CONCLUSION

While the articles that a given user will read can be predicted on an individual scale, it was found that this method resulted in rather poor accuracy due to a lack of data for each individual. As such, the users were clustered together by their reading habits, and these clusters were then used to determine the whether a given article would be recommended to a given user. Two methods of determining this recommendation were used, majority voting and super-users. With majority voting, each article was tested on each user within a cluster, and the 'votes' were tallied, with the majority vote determining the recommendation. For the super-users, that data from all the users within a cluster were combined and treated as if one user had read all of the corresponding articles. The given article was then tested on this super-user, with the result being the recommendation given. As discussed above, the utilization of super-users via clusters was found to provide the most accurate predictions.

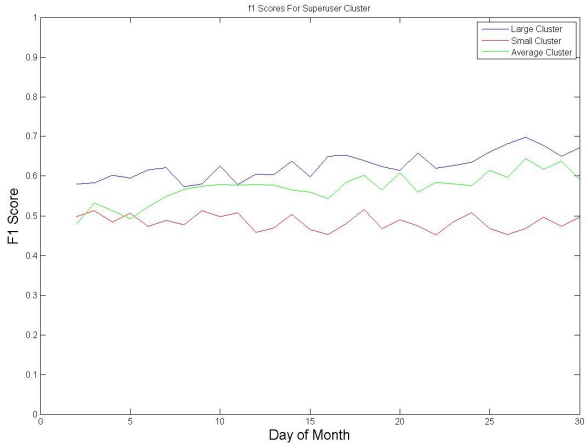

Figure 7: f1 Scores for Super-user Clusters - This plot shows several f1 scores of clusters used for the super-users over the month of August. The blue curve is of a large cluster, the red of a small cluster, and the green is the mean f1 score of all clusters. Similar to the f1 scores from majority voting, the large cluster has a better daily f1 score than the small.