TF-IDF CHAT USER CLUSTERING

KEVIN MALINAK, JONATHAN SWENSON
MALINAK1@STANFORD.EDU, SWENSONJ@STANFORD.EDU
SUID: 05720218, 05713257

DEC 14, 2013

Introduction

Our project applies a k-means clustering algorithm to pair users on Chatous, a randomized chat site. The user-pairing system currently in place utilizes a weighted randomized algorithm which takes into account given profile information about the users, including age, sex, region, and interests. However, the algorithm overlooks a wealth of information inherent in the actual raw text of the chats. By analyzing the past chat history of a certain user, we can provide a cluster of users that share similar lexical characteristics, and we expect that pairing users who user similar vocabulary will improve overall chat quality.

Input Data

The founders of Chatous have provided us with a dataset of around 9 million chats between users with some metadata about each chat, such as which user disconnected, the number of chats that each user sent, whether or not the chat was reported and who reported it, the time that the two users spent chatting as well as whether or not the chat was reported. As Chatous is interested in protecting the privacy of their users, they have redacted the actual text of the chat, but simply left us with the word frequencies for the chat where each of the words has a global unique ID that they can use to look up words if we need them to.

Model

We implement a TF-IDF (Text Frequency - Inverse Document Frequency) approach to weight a user's vocabulary according to a global vocabulary[1]. With this model, we can examine how frequent certain words appear in a user's vocabulary and can determine what may be important to that user. To do this, we form a "feature vector" of the user's vocabulary and then weight each of the features differently with a TF-IDF model, which gives a high weight to words that are used frequently by the user, but not used as frequently overall (over all the user's vocabularies). This weighting allows us to emphasize more individualized linguistic choices, as opposed to universally used words such as "and", "the", and "you", etc.

[1]http://nlp.stanford.edu/IR-book/html/htmledition/tf-idf-weighting-1.html

To define in specific terms, in our input data we are given conversations between two users. Each conversation contains user 1's ID, user 2's ID, and a set of words and the frequency of each word used by each user in that conversation. First, for each user, we sum the word frequencies over all documents associated with that user. This results in a vocabulary for each user which maps a word to the user's total frequency of that word. Then, we calculate the document frequency df_t, which denotes the number of documents in which the word t occurs. This helps us compute the inverse document frequency, which is defined: $idf_t = \log \frac{N}{df_t}$, where N is the total number of documents. Finally, for each user, we iterate through their used words and multiply their frequency, denoted t_f, by their respective idf values. Thus, our final model consists of a feature vector $\phi(x)$ for each user, where x is the input data and $\phi(x)$ is a vector where the ith element is the $tf - idf$ weight of that word.

Algorithm

Basic Structure

The basic structure of the algorithm that we are implementing is the k-means clustering algorithm as given to us in class. We use the weighted feature vectors as calculated in the above model and cluster them around k centroids. However, we deviate from the standard k-means in the objective function which we wish to optimize. Instead of measuring and minimizing the Euclidean distance between a feature vector and the centroids, we instead attempt to maximize the **cosine similarity** of the vectors; that is, we maximize the function $\cos\theta$ where θ is the calculated angle between the feature vector and the centroid vector[2]. Since our feature vectors are in positive space (ie, words cannot be used negative times), this becomes a useful metric, as the outcome is bounded in $[0, 1]$. The cosine similarity also has an advantage over the Euclidean distance because if two particular users have similar vocabulary, but use the words in significantly different frequencies, we want them to be placed in the same cluster; however, the Euclidean distance function would potentially place them in different clusters.

Parameter Selection

We optimized our algorithm by varying the following parameters until we converged towards an optimum:

- The number of clusters: $k = 10$. With fewer clusters, we found that "super-clusters" formed, which could be further separated with increasing performance, because we found several "sub-clusters" which contained relatively little inter-cluster conversations. The super-clusters. With more clusters, however, our algorithm resulted in "sub-clusters" which could be coalesced, again with increasing performance.
- The size of the training and testing datasets: $n = 100,000$. We need only consider a subset of the 9 million chats to test the accuracy of our algorithm; furthermore, due to the inherently large dimension of our feature vector (representing the entire vocabulary of a single user), the runtime

[2] http://pyevolve.sourceforge.net/wordpress/?p=2497

becomes unmanageably long. For the purpose of testing our algorithm, $100,000$ conversations presents a sufficient clustering to achieve reasonable results.

- The number of k-means iterations: $i = 15$. We found that, given randomized initial centroids, our algorithm tended to converge after a maximum of 15 iterations.
- Centroid initialization: $\mu_i = \frac{1}{m}\sum_{j=1}^{m} \phi_j(x)$. As an initial "guess" for the centroids, we randomly select $m = 500$ feature vectors and set μ_i equal to their average. This results in decreased runtime, as our initializations are presumably closer to their "true" values than if we had set their values completely randomly.

Results and Analysis

Evaluation Metric

The metric by which we judge the success of our algorithm is the ratio of average conversation length within a cluster (intra-cluster) to the average conversation length between clusters (inter-cluster). We chose this metric because we set out to increase conversation quality, and it is safe to assume that two people who are enjoying a high-quality conversation will tend to have longer conversations. Our clustering algorithm is then successful if it discovers clusters of users whose intra-cluster conversation length is **high**, but inter-cluster length is **low**; this signifies that users within a cluster tend to have higher quality conversations with other users in that cluster, as opposed to users in other clusters.

Analysis

By the metric of conversation lengths, our algorithm was successful in increasing chat quality over the clusters. We first trained our clusters on a randomly selected subset of $100,000$ conversations; then, using the resulting centroid values, we clustered a mutually exclusive test set of $100,000$ conversations. In both the training and test sets, for each cluster we calculated the average conversation lengths within itself (intra-cluster) and from itself to each other cluster. The results, given in Table 1 and Table 2, demonstrate a significant increase in intra-cluster conversation length versus inter-cluster conversation length over all the clusters, and in both the training and test sets. Through clusters 1-9, the increase in conversation length can be seen clearly over the given users; however, cluster 0 presented a more interesting case, which we will address in the following section. Additionally, the test data produced two clusters (4 and 9) which had no intra-cluster conversations; this is to be expected, as cluster 4 has 1 user (and thus cannot have a conversation with itself) and cluster 9 has 2 users (and it is entirely possible for two randomly selected users to not have had a conversation yet).

The Outlier Cluster (Cluster 0). Despite the clear improvement in conversation length in the other clusters, we encountered a cluster which did not match the others and therefore presented a contradiction to our theory. Cluster 0 contained a very large number of users who had relatively low conversation lengths;

Table 1. k-means Clustering Results on Train Data

Cluster	Number of Users per Cluster	Intra-Cluster Conversation Length	Average Inter-Cluster Conversation Length
0	11118	0.645	1.283
1	944	19.832	1.882
2	1434	50.542	2.228
3	847	16.556	1.771
4	91	163.917	1.968
5	4462	4.317	1.938
6	1974	23.718	2.143
7	416	225.213	2.190
8	114	108.500	1.847
9	99	64.479	1.855
Average	2150	67.772	1.911

Table 2. k-means Clustering Results on Test Data

Cluster	Number of Users per Cluster	Intra-Cluster Conversation Length	Average Inter-Cluster Conversation Length
0	16349	0.658	1.309
1	260	16.821	2.002
2	331	38.630	1.926
3	218	15.573	1.603
4	1	N/A	1.855
5	3655	4.330	1.918
6	558	20.0	2.166
7	203	185.927	2.241
8	3	202.911	1.799
9	2	N/A	1.764
Average		60.606	1.858

more importantly, this cluster exhibited a **lower** intra-cluster conversation length than inter-cluster length. In layman's terms, we found a cluster of users who did not like to talk to other users in the same cluster, but liked to talk to users outside of that cluster. Our first hypothesis for this phenomenon was that this cluster was comprised of male or female users who did not want to talk to the same sex, or that it was comprised of users from a specific region; however, an analysis of the metadata showed that neither of these cases were true. We compared Cluster 0 to Cluster 7, which we decided to be roughly indicative of the other clusters, and found that the distributions of gender and region were roughly similar (see Tables 3 and 4). Thus, a deeper analysis of the cause of Cluster 0's characteristics is needed. Unfortunately, due to the fact that the original words used by our users has been redacted, it is difficult to produce meaningful results; however, our prevailing hypothesis is that Cluster 0 contains users who are "shy" and do not tend to initiate conversations. When two shy users are placed in a chat, they wait for the other user to reply; when they do not, they cancel the conversation and move on. However, when this user is paired with a user from a different cluster, they will tend to carry on a longer conversation.

Conclusion

We present a finalized process for pairing users using the k-means clustering system:

Table 3. Comparison of Cluster 0 and Cluster 7 by Gender

	Training Data			Test Data		
Cluster	% Male	% Female	% Not Specified	% Male	% Female	% Not Specified
0	53.581	42.510	3.909	53.681	42.456	3.863
7	52.217	43.842	3.941	56.853	38.579	4.668

Table 4. Comparison of Cluster 0 and Cluster 7 by Region

	Training Data					Test Data				
Cluster	% US	% India	% UK	% Canada	% Other	% US	% India	% UK	% Canada	% Other
0	40.403	12.117	7.149	4.227	36.104	40.155	12.258	7.150	5.198	35.239
7	43.147	14.213	4.061	6.091	32.488	46.798	9.852	5.911	3.941	33.498

(1) Train the clusters using the k-means algorithm with cosine similarity on a subset of the dataset.

(2) Given a new user, calculate the closest centroid using cosine similarity.

   (a) If the user conforms to the outlier cluster, pair with a user from a different cluster.

   (b) Else, pair with another user from the same cluster.

Using this process, we have empirically shown an increase in conversation lengths over the given dataset. Users will tend to have longer conversations, which is correlated with higher chat quality and a better overall experience with the Chatous application.

Future Analysis

As noted, the redaction of used words in the dataset prevented us from a deeper analysis of our results. With this extra information, we may be able to classify and generalize each cluster based on the used words; for instance, the most frequently used words for each cluster could possibly correlate with topics of the conversations within that cluster, or we could determine the root cause of the outlier cluster's appearance in the data. Furthermore, access to true user vocabulary could help to determine bots spamming the site – a user spamming a URL or a particular product name is much more likely to be a bot.