

# A Prediction System For Conversation Likeability On Anonymous Chat Networks

Andrei Bajenov  
abajenov@stanford.edu

Saila Talagadeevi  
saila@stanford.edu

**Abstract**—We implement and evaluate a prediction system for conversation likeability between a set of people on an anonymous chat network using Support Vector Machines, Logistic Regression, and k-means clustering algorithms. Here we use k-means clustering for unsupervised feature learning which will be fed to supervised classification methods. Our algorithms draw principles from supervised and unsupervised machine learning.

**Index Terms**—k-means, SVM, Logistic Regression

## I. INTRODUCTION

Chatous is a text-based, one on one, anonymous chat network where users can have text-based conversation with random strangers. Chatous has over 2.5 million unique users across 250 countries. Each user in Chatous has a profile where he/she can specify a few simple facts such as gender, age, and location. Each user can perform actions such as starting a new conversation, friending, reporting a conversation, and ending a conversation.

When a user comes online in the Chatous network, he/she is presented with the most probable chat buddy from a pool of available online users. Our work aims to improve the algorithm used to select a buddy that will result in the highest conversation likeability.

## II. PRIOR WORK

The creators of Chatous have done some work in trying to predict conversation likeability. They trained a logistic regression model that was able to give them an edge over random matching. Their successful features included gender, and users' history of previous interactions with mutual buddies.

We would like to build on their work by extending the feature set and trying different algorithms. One thing that was not incorporated into their models was the words used by users in prior conversations. We claim that words used in prior chats are good signal. We can use this information to better match users with similar interests. We would also like to add additional signal about a user's chatting behavior. For example, a user's actions in the past time buckets (such as ignoring chats in the past hour) may give a strong indication to what he/she may do next.

## III. DATA

Chatous provided us with large data sets to build our ML models:

- **Chat history (9,050,713 entries):** Contains information about each chat. It includes the length of conversation, the number of lines each user wrote, word frequencies

for each participant, which user disconnected, whether a friendship was established, and whether a report was made.

- **User profiles (332,888 entries):** Contains user-specific information. User id, age, gender, location, and word frequencies in the "about me" section.

For privacy reasons, words are masked in the data set. Each word is mapped to an id, and frequencies are given for each word id. A list of stop words is also provided so that we can ignore them while clustering.

## IV. DATA PROCESSING AND FEATURE GENERATION

### A. Labels

Close to 75% of conversations on Chatous are empty. This means that 75% of users are matched up with a buddy who disconnects right away. We think this is very poor user experience and believe that if we can match up users who are likely to begin a conversation, the overall user satisfaction will improve.

As such, we label a conversation as good if it satisfies **all** of the following:

- Neither user reports the conversation
- Both users write at least one line
- The conversation length is longer than 15 seconds **OR** the two users friend each-other **OR** both users choose to leave the conversation open

This labeling is not perfect, since it may label very short conversations as good. However, since so many of the conversations are empty, this gives us a decent indicator that a conversation could turn into a satisfying one. Moreover, this labeling will help identify users who are generally bad actors, or who are usually inactive.

### B. Features

Every conversation involves two users. In order to predict whether a conversation will be good, we extract features for both users and combine them into one feature set. Below is a full list of features that we are working with. We discuss how we prune this feature set later.

- Users' gender and whether or not they have the same gender
- Users' age and age difference
- Whether or not they are in the same country or region
- Chat history features (we have multiple sets of these for 1 minute, one hour, one day, and all time buckets):

- How many times a user was reported and reported someone
- How many times a user ended a conversation and was left by someone in a conversation
- Numbers giving the types of conversations a user was involved in. For example, empty conversations and one-way conversations
- Total number of conversations
- Average conversation lengths
- The word cluster a user belongs to (determined via the k-means algorithm discussed later).

Some of the features above are for a single user and some are for pairs of users. Where it makes sense, we average the individual features for both users. For example, we average the average conversation length for the two users, and include it as a feature.

Overall we get about 1100 total features from this (excluding features related to user clusters).

## V. ALGORITHM STEPS

Our goal is to assign weights to pairs of users, to indicate the likelihood of them having a conversation.

We first use logistic regression to train a model on the set of all features. Using the model learned by the logistic regression, we identify the features that have a strong correlation to the outputs and use only those features in subsequent training.

To gain signal from words used by user, we use k-means to cluster users. Once we have assigned every user to a cluster, we feed this information as features into the original classifier and analyze the results.

Having identified a set of good features, we train an SVM with various kernels to see if we get better results.

### A. Feature Selection

We train a logistic regression model as an easy way to identify features that have a strong correlation to the output. Since we have over 1,000 features, this is a quick and easy way to identify bad features. To identify features that have weak correlation to labels, we look at the  $\theta$  learned by the logistic regression and pick the values with lowest magnitude. We are able to do this since we scale our feature vector.

### B. Support Vector Machines

We attempt to use SVMs for training because it gives us the flexibility of using non-linear kernels. We experiment with various features to the SVM as well as with different kernels.

### C. k-means clustering for feature learning

We use k-means clustering as a feature-learning step for our supervised learning algorithms. In this method, we perform clustering on a chat word similarity matrix to identify and label users who tend to chat about similar interests and topics. For each labeled sample, the distance to each of the k learned cluster centroids is computed to induce k extra features for the sample. The feature can be a Boolean with value 1 for the closest centroid and -1 for other clusters.

First we generate the Jaccard Similarity Matrix to feed into the k-means clustering algorithm. The Jaccard similarity measures the similarity between two sets. It is defined as the size of the intersection divided by the size of the union of the sets:  $J(A, B) = \frac{|A \cap B|}{|A \cup B|}$ . We then normalize the similarities.

Algorithm steps:

- 1) We build common word vectors from raw data. We also build an adjacency matrix with users vs. words.
- 2) We build a similarity matrix from the adjacency matrix using Jaccard similarity.
- 3) We run k-means clustering on the similarity matrix. kmeans will cluster users who used similar words (words related to sports, movies, music etc.). The underlying assumption is that users who talk about a particular topic will most likely chat again about the same topics and will like chatting with people who like to chat about those topics.
- 4) We use a similar procedure to generate user clusters based on words in the "About Me" profile section.

## VI. RESULTS AND FINDINGS

### A. Feature Strength

After running logistic regression on a scaled feature set, we identified the following top features:

- 1) Gender is by far the strongest predictor for a positive chat. Chats of opposite gender and chats with two female users, tend to be non-empty and last longer.
- 2) A high age difference has a negative correlation to a positive outcome. The higher the age of either user, the lower the likelihood of having a conversation.
- 3) The one hour chat history bucket seems to be a strong predictor of future behavior. Averages between two users of the number of empty chats they had, how many times they were ignored, how many times they engaged in a conversation, etc.
- 4) The average conversation lengths of two users in the all time and 1 hour bucket have a positive correlation on the outcome.

Features for other time buckets, country and region features, and reporting-related features play a smaller role in the prediction.

### B. Logistic Regression Performance

We trained a Logistic Regression model as a quick way to study the data and select good features. It gave us a sense of roughly how many data samples are needed for training given our feature sizes.

Initially we trained a Logistic Regression on 100k chat samples. We used all 1000+ features described earlier (excluding the clustering features), and got **86.36%** cross validation accuracy for 10-fold cross validation.

To get an idea of what to do in order to improve the accuracy, we ran some diagnostics for the classifier. Fig. 1 shows a diagnostic we ran, where we varied the number of samples for our test and training sets, and plotted that against

the prediction error. The plot showed that a 100k sample size is large enough. In fact, we found that using more than 10k samples for training does not have a significant impact on classifier performance.

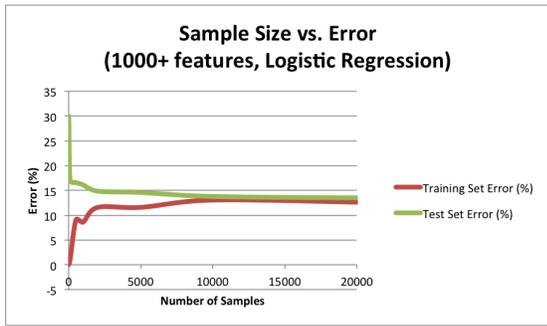


Fig. 1. Sample Size vs. Error for Logistic Regression with 1000+ features.

Fig. 1 also showed us that we were not overfitting, since the training and test data had similar errors after 10k samples. This confirmed that adding additional features (such as word clusters), or using a different classifier (such as SVM) might help overall accuracy.

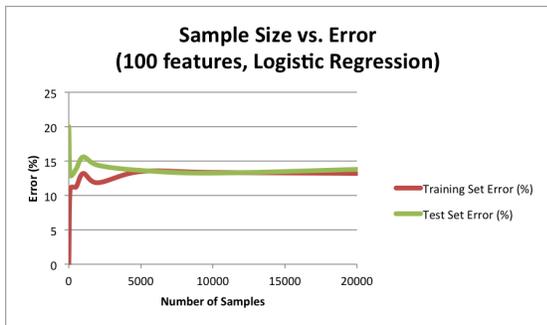


Fig. 2. Sample Size vs. Error for Logistic Regression with 100 features.

Since 1000+ features is quite a lot, and significantly slows down training time for SVM (which we train later), we decided to prune our feature set to top 100 features (based on the analysis from the previous section). With only 100 features, we got a cross-validation accuracy of **86.34%**. The small drop in accuracy is not a high cost, given the benefits of using fewer features. Fig. 2 shows a similar diagnostic for the Logistic Regression classifier with only 100 features. We see that it requires fewer training samples to obtain similar accuracy results.

In order to verify the usefulness of our accuracy metrics, we plotted precision/recall curves for positive and negative classification. Fig. 3 and Fig. 4 show these curves for Logistic Regression (and also SVM which we discuss later) using 100 features. Observe that we are doing much better than random matching in both cases. The random lines were generated based on the ratio of positive and negative labels. We have about 15.9% positive labels and 84.1% negative labels. These figures helped us confirm that the classifier does not

just predict a single class all the time, which validated the usefulness of our accuracy metrics.

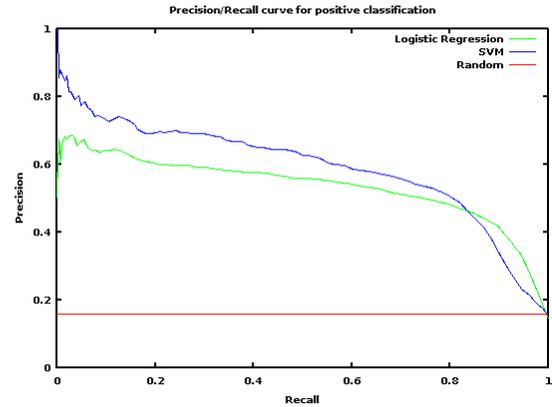


Fig. 3. Precision/Recall curve for positive classification.

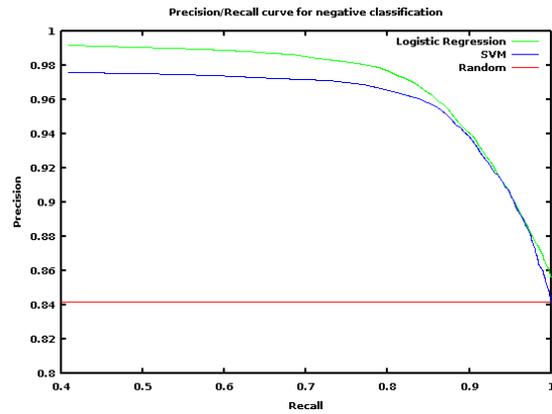


Fig. 4. Precision/Recall curve for negative classification.

### C. SVM Performance

We used Logistic Regression to study the data and select top features. We observed a high bias, so using an SVM with a non-linear kernel had potential to improve accuracy.

We trained an SVM with an RBF kernel. It had about 1% higher (**87.29%**) cross-validation accuracy than Logistic Regression. It also showed better precision for positive classification (see Fig. 3). Unfortunately it showed lower precision for negative classification. However, since we value matching good users more, higher precision on positive chats is better.

We experimented with different values for C (cost) during training and found that 512 works best. We also tried alternative kernels, and saw that the polynomial and RBF kernels have the best performance, but the RBF kernel has a slight edge (0.03% higher cross-validation accuracy).

We also ran a sample size vs. error diagnostics for SVM. Fig. 5 shows the behavior of the SVM model as we vary the training set size. It requires more data to get good performance, but otherwise looks very similar to that of Logistic Regression. There is also a higher gap between training and test error at

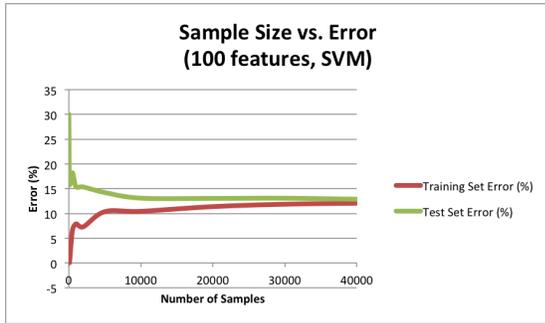


Fig. 5. Sample Size vs. Error for SVM with 100 features.

higher sample sizes, suggesting that training with more data may further improve accuracy.

Overall we found that SVM with an RBF kernel shows higher accuracy, but takes much longer to train and slightly longer to predict than Logistic Regression.

#### D. Clustering Performance

Having trained SVM and Logistic Regression on a base set of features, the next step was to add signal from words used in previous chats.

We found that the adjacency matrix of chat words turns out to be in several hundreds of megabytes if we consider every word and every user who participated in the chats. After analyzing the data, we found that most words have low frequency of occurrence across chats. Also a lot of users do not chat a lot. These low frequency users and words prove to be irrelevant for clustering and hence for prediction. After pruning the data set we reduced the adjacency matrix to under 30 megabytes. We did additional fine tuning by removing stop words from the word vectors to improve the cluster quality and word relevance to user interests. Removing stop words significantly increased our cluster quality.

We were then able to use k-means to cluster users into 20 clusters. For every cluster we added a feature to our SVM. We labeled the feature +1 if one of the users in the chat is from that cluster, and -1 otherwise. We added another feature that had a positive label if both users were from the same cluster. This increased the SVM cross-validation accuracy by 0.65% (**87.94%**).

We also tried to add clusters for words in the "About Me" section of users' profiles. We found that clustering against these vectors had surprisingly little effect on chat likeability predictions. We decided to remove these clusters from our feature list.

## VII. CONCLUSION

We built a system that predicts if a pair of users will have a successful chat with up to **87.94%** accuracy and good precision/recall performance. Since Chatous is an anonymous chat network, its success is very much dependent on pairing a user that comes online with the most likeable user that is available at that moment. Accuracy as well as speed of prediction makes all the difference in a user's experience.

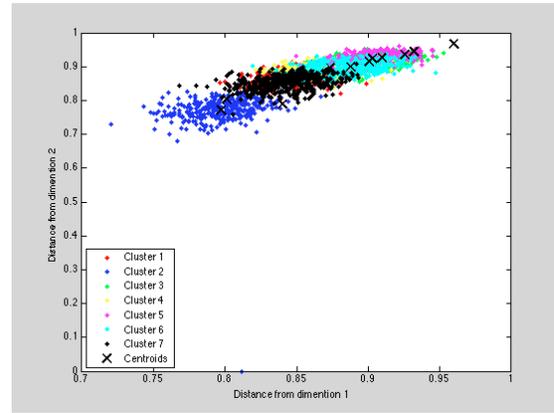


Fig. 6. k-means 2-D plot with first 7 clusters

Even with very noisy data, we were able to identify several strong features. We saw that gender is by far the strongest predictor of a positive chat. Chats of opposite gender tend to last longer and have a higher chance of being non-empty. We also observed that users' previous chatting behavior is a strong predictor of future actions.

We also looked into using words from previous chats as a signal for our predictions. We ran k-means clustering on word vectors from previous chats to cluster users into groups. We then fed these clusters as features to our classifiers. We saw a 0.65% improvement in accuracy. We also observed that the words used in the "About Me" section aren't very useful for predicting a successful chat.

For supervised learning, we used Logistic Regression and SVM. We found that SVM with an RBF kernel has slightly better cross-validation accuracy than Logistic Regression, but the runtime performance is worse.

We believe that integrating our prediction system should improve the overall user experience in Chatous.

## ACKNOWLEDGMENT

We thank Chatous and Kevin Guo for providing the data set and spending time explaining it.

## REFERENCES

- [1] Csurka, Gabriella; Dance, Christopher C.; Fan, Lixin; Willamowski, Jutta; Bray, Cdric (2004) "Visual categorization with bags of keypoints". ECCV Workshop on Statistical Learning in Computer Vision
- [2] Coates, Adam; Ng, Andrew Y. (2012). "Learning feature representations with k-means"