John Kim
Michael Nussbaum
Jeffrey Schecter

Twitter Predictor

contact: twitterpredictor@gmail.com
codebase: https://github.com/michaelnussbaum08/Twitter-Predictor

Our project attempts to predict the frequencies of words among communities of Twitter users: given the tweets of a population up to day i, we want to know how often a word will occur among that poopulation on day i+1. This sort of information on what topics in a community are becoming more or less popular has obvious value for advertisers and news junkies, as well as academic import. For instance, word frequency prediction would be of value to linguists investigating the spread of novel forms through a community.

Raw data consisted of populations of twitter users and their 200 most recent posts. We pulled communities of highly interconnected users starting with single users as seeds, creating a 200 member training community based around the user lizardbill and a 199 member testing community from the user davechapman.

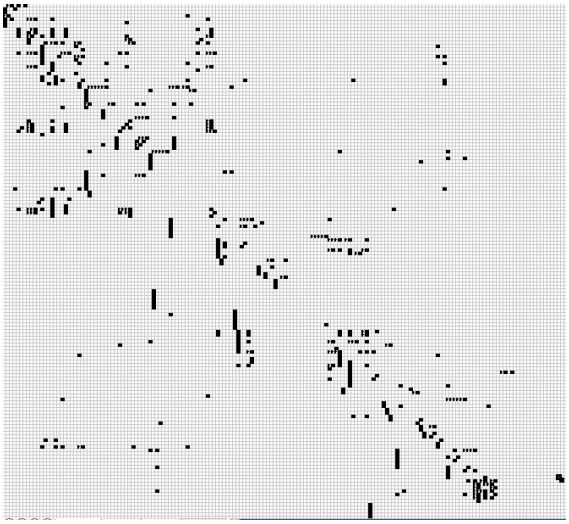

figure 1—Followers in the lizardbill community. Black cells indicate the user represented by the column follows the user represented by the row.

Communities were then processed to create datasets wherein each instance represents a word. The tweets of the population as well as the Hoosier corpus, a machine readable dictionary often used in computational linguistics, were used to calculate attribute values. Continuous attributes were the frequency of the word in the population on days i+1, i, and i-1, the total frequency of the word in the population up to and including day i, the frequency of the word as reported by the Hoosier corpus, the experimentally determined familiarity rating of the word given in the Hoosier corpus, and a predicted exposure rating computed via network analysis.

Nominal attributes were the word's presence or absence in the Hoosier corpus, the word's status as a hashtag (words beginning with # indicating the topic of a twitter post), and the word's status as an atreply (words starting with @ indicating the intended reader of a post).

Expected exposure for word *w* on day *d*, written $E_{w,d}$ is defined as the total number of times that we expect community members to read *w* on *d* in the posts of other community members. We assume that each user reads every tweet posted by a user that he follows exactly once. We further assume that reading a word makes a user more likely to write that word in turn. Let the parameter rho be the "remention rate"—the probability that reading a word inspires a user to mention that word in his own posts within a day. For our calculations, we used a rho of .01.

To find $E_{w,d}$, start with the follower matrix, F. F_{ij} is equal to 1 if user *j* follows user *i*, else zero. Let **v** be a vector whose i^{th} element is the number of times that user *i* mentions word *w* on day *d*. Observe that **v** • F produces a vector whose j^{th} element is the number of times that user *j* will read word *w* on day *d*, provided each user reads each post by someone he follows one time.

Observe also that in the friend matrix elevated to the n^{th} power, F^n, index *i, j* will be the number of unique "who-follows-who" paths through the community from user *j* to user *i*. The j^{th} element of the vector (**v** • F^n) x (rho^n-1) is therefore equal to the number of times that user *j* will read *w* on *d*+1 just because people he follows have been inspired to write *w* through paths of length n-1. For instance, for n=2, index *j* will be the number of times that *j* reads *w* because people he follows were inspired to write *w* by original authors of *w*. To find the number of times that users read *w* because of inspiration paths of any length, find:

$$\mathbf{u} = \mathbf{v} \bullet \sum_{n=1 \text{ to } \infty} (F^n)(rho^n-1)$$

Summing along the indices of **u** produces the total number of times that community members should be expected to read *w* within the next day. In practice, we used a maximum n of 10.

Prediction of frequencies on day i+1 was accomplished via regression with a support vector machine, using libSVM 3.0. Grid search was used to find optimal c and gamma parameters. Learners were trained on the lizaardbill community and tested on the davechapman community. SVM regression was attempted with the linear, radial basis, polynomial, and sigmoid functions as kernels, with nu and epsilon support vector regression, and was performed with and without the expected exposure data. The best results were achieved with network analysis, nu SVR, and a polynomial kernel, producing a mean sum squared error of 0.163374 and a squared correlation coefficient of 0.519282. This represents a ~0.0043 improvement over the best regressor which did not include network analysis data. For every kernel other than the sigmoid function (which had the overall worst performance), better results were achieved with expected exposure data than without.

Mean squared errors for SVM generated models were compared against two baselines. In the first baseline, the frequency of every word on day i + 1 was taken to be the average frequency on day i. In the second baseline, frequencies on day i + 1 for word *w* were assumed to not change from day i. Every model was able to outperform both baseline measures.

		Without network analysis		With network analysis	
Type of SVR	Kernel	Mean squared error	R squared	Mean squared error	R squared
epsilon	linear	0.21097	0.464686	0.203477	0.469620
epsilon	polynomial	0.165488	0.483638	0.148875	0.512185
nu	linear	0.209521	0.460854	0.206802	0.463235
nu	polynomial	0.181265	0.492836	0.163374	0.519282
epsilon	RBF	0.181273	0.514984	0.178821	0.515204
epsilon	sigmoid	0.247588	0.419747	0.314687	0.266598
nu	RBF	0.180704	0.510002	0.178232	0.511703
nu	sigmoid	0.238834	0.422641	0.275716	0.331316
BASELINE, average		1.387903			
BASELINE, day i = day i + 1		0.530791			

figure 1—Error measures and squared correlation coefficients by SVM and kernel type

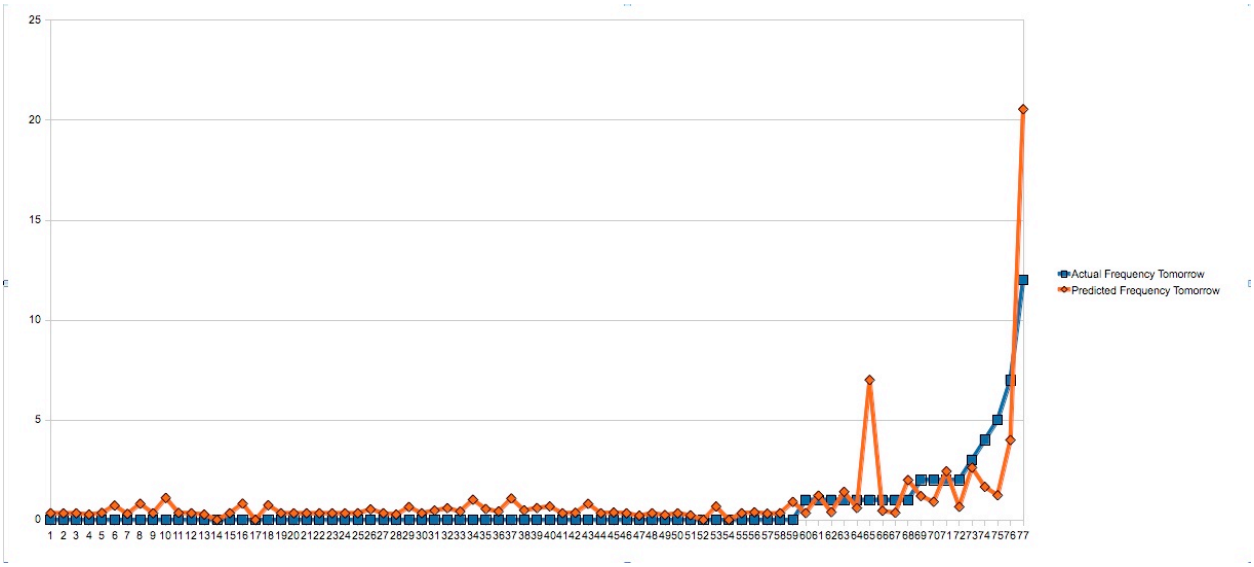

figure 1—Predicted and actual frequencies on day i+1. Predicted frequencies were generated with the best SVM model. Words are arranged in ascending order of actual frequency on day i+1 along the x axis; frequencies are shown on the y axis.