A machine learning approach to qualitative analysis in behavioral research.

Authors: Andrew Reece & Colin Bosma

R packages used: boot, glmnet, Matrix, SnowballC, tm
Dataset: "Innovation Study" (private); PI: Ellen Langer

Introduction

Our research lab recently conducted a study which took qualitative (ie. written response) measures as its primary outcome measure. Our goal was to assess whether preconceived ideas about object utility interfere with innovative thinking.

In this study, we described a number of failed consumer products to participants. 121 participants were randomly assigned to one of three groups (A, B, and C). Group A members were told what the product was intended to be, and that it had failed. Then they were asked what they could do with such a product. Group B members were only told what the product was intended to be, and then asked for alternate uses. Group C members were only given a description of the products physical properties (ie. shape, weight, contour, etc.), and then asked what it could be used for.

Our working hypothesis was that Group C's written responses would be judged most innovative, as they were given the fewest preconceived notions as to what the product had originally been intended for. For the same reasons, Group B might perform better than Group A on the same measure. Written responses were rated by a rater who was blind to condition, on a 1-10 scale (not innovative to very innovative). (NB: Neither one of us actually designed this study, which can fairly be criticized as having a number of methodological flaws. Nevertheless, we can still perform the analyses we describe below to understand the study data better.)

Qualitative measures are frequently avoided in psychological research, largely because accurate representation of findings requires coding of the qualitative data into numeric values, and coding standards vary widely (and are thus looked upon with some suspicion).
The standard approach to evaluating qualitative data is to have multiple raters evaluate the data, and then compute some degree of inter-rater reliability.

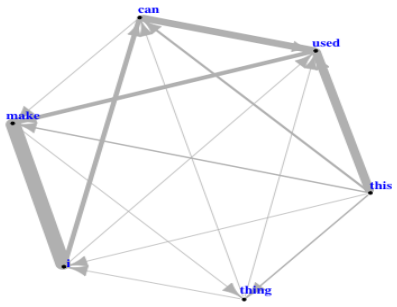
There are a number of points along this analytical path which are open to criticism, including the rationale for the coding schema used, as well as what constitutes acceptable intra-class correlation coefficients among raters. In contrast, statistical learning methods for text analysis generally operate on low-level features like term frequency and document length, rather than the higher-level semantic evaluation done by human coders. As such, a purely algorithmic approach may fall short in comparison to human analysis in terms of nuance and connotation, but it has the virtue of being unaffected by precisely those aspects of text which are quite

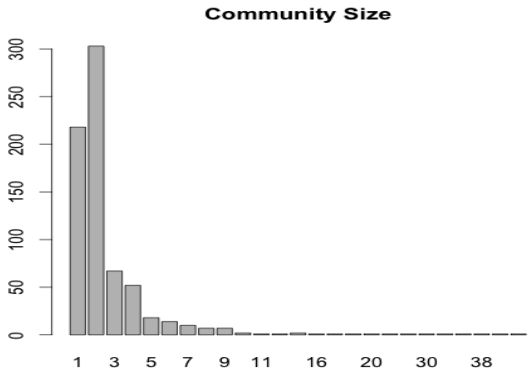
open to interpretation (and, transitively, human misevaluation). Here, we apply a statistical learning approach to first model the data and then conduct null hypothesis testing, as a means of introducing a greater degree of objectivity in qualitative analysis.

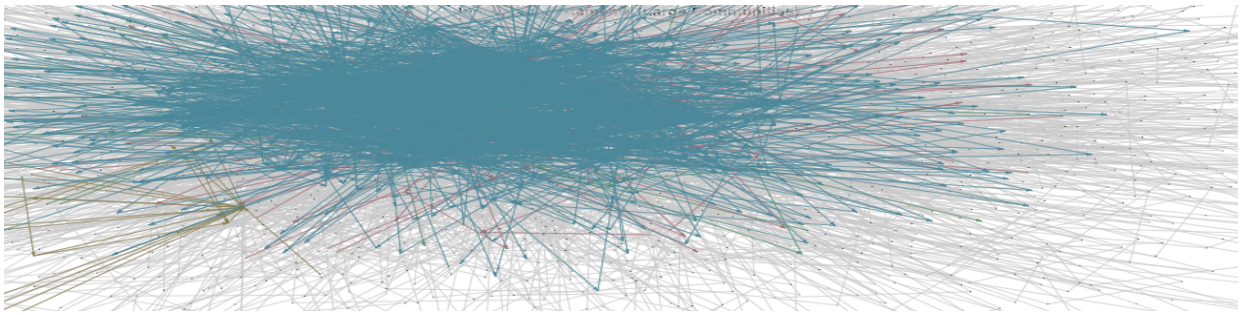
Method

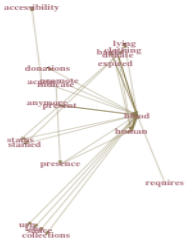
1. Community detection

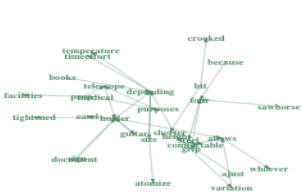
We had initially thought to base our work on the community detection framework. We encountered two problems, however, that required us to search for a different approach. In network-graphs.R, we modified the example community detection code to generate linguistic network graphs of our Innovation Study data. Below are graphs generated of “largest clique”, frequency distribution of community size, and finally the fully network graph with communities overlaid in color:

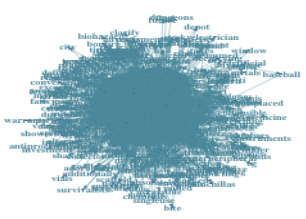












Whereas with the GOP data, the linguistic sub-groups that fell out of the community detection algorithm were often clearly identifiable as political factions (ie. religious right, fiscal conservatives, etc.), the sub-groups we observed in the innovation data were much harder to categorize. We also ran the same analyses for the writings within each experimental condition, and while the communities that emerged were definitely different, the differences were impossible to explain in any meaningful way.

An additional problem is that the linguistic communities are not clearly associated with experimental condition. While it is possible to include metadata in a text corpus (using the tm package), it doesn't carry over into the matrix of word pairs used to create the nodes and edges of the network. Even if the linguistic communities discovered in the dataset were clearly organized around semantic themes, it remains unclear as to just how we might go about assessing their relationship to the experimental group assignments, which is our ultimate aim in this analysis. (A bit more on that later.)

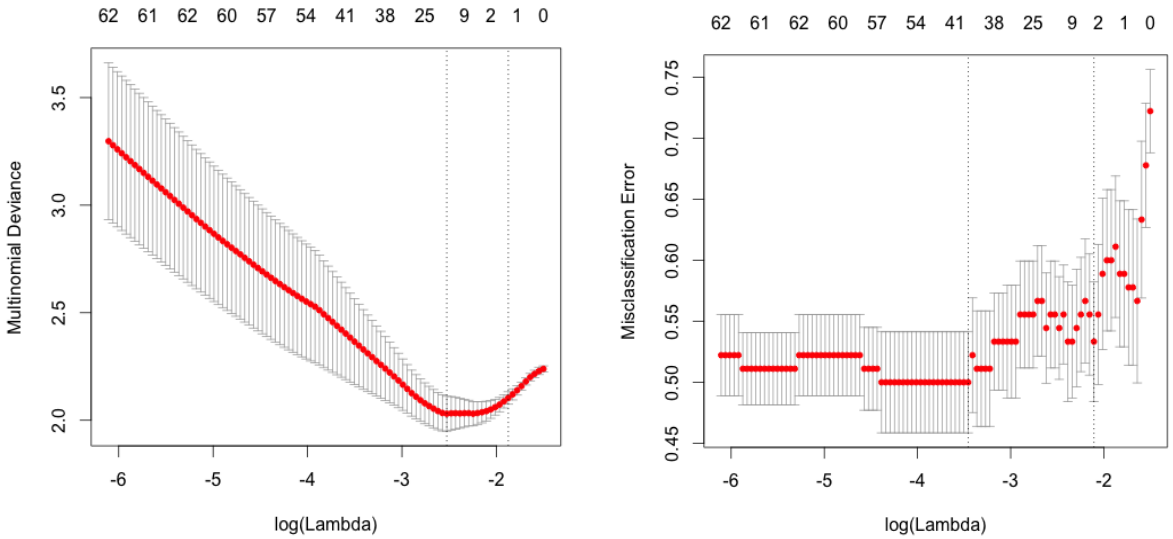
2. Words-as-predictors and dimensional reduction

Our second attempt took a more conventional approach towards text analysis. Using the `tm` package, we converted the entire dataset into a document-term matrix (DTM), which is essentially a sparse matrix of term-weights per document. (We experimented with two forms of weighting - we used a standard term-frequency, as well as tf-idf, results reported below.) This resulted in a 121x2565 matrix (121 subjects' documents and 2565 unique terms in the corpus). We then appended the original vector of experimental condition assignments to the matrix.

A DTM is essentially a sparse matrix, and the `tm` package actually organizes DTMs in the standard triplet format used in many sparse matrix compression schema. But we formally converted our DTM to a sparse matrix object using the Matrix package, as the packages we used for analysis later on required proper sparse matrix classes as input.

With this format, we were able to treat each term that appeared across the entire corpus as a predictive feature, with the condition assignment vector as our response variable. As our response variable has three discrete classes, we can apply a multinomial logistic regression. At this point, however, our feature set (P=2565) was much larger than our sample (N=121), and so dimensional reduction of some sort was called for.

We addressed this issue using the `glmnet` package to perform a LASSO regularization, using the multinomial classifier setting. We split our data into train and test sets, at a 3:1 ratio, and then used 10-fold cross-validation to find the optimal penalty parameter (λ) to minimize our loss function. We tuned λ using both deviance and misclassification rates as the evaluation metric, with the resulting outcomes:



NB: We compared the models resulting from both loss metrics, and they were similar across multiple comparisons within a percentage point or so, so we ran later analyses using only the deviance-loss models.

Once we'd fit a model from glmnet, we used it to make predictions on our test datasets. The prediction function uses λ at either the global loss minimum, or at 1.0SE away from the loss minimum (in the less restrictive direction). We used the λ value at the global minimum (which evaluates $\log(\lambda)$ at roughly 2.5). Once we had generated a vector of predictions on our test data, we were able to evaluate the model's accuracy. We noticed quite a bit of variance in accuracy across separate model fittings, even with the cross-validation built in; correct classification rates ranged anywhere from about 35-80%. To be safe, we fitted 100 cross-validated models, each with different randomized train/test sets, and then averaged the accuracy across all the models. Average correct classification rates hovered around 52% when using term-frequency weighting, and around 54% when using tf-idf weighting[1].

Predicting condition assignment at random with a three-level factor should result in classification accuracy of only 33%, by comparison. Given the large variance in classification rates, however, we decided to generate a null distribution by permuting our model's outputted

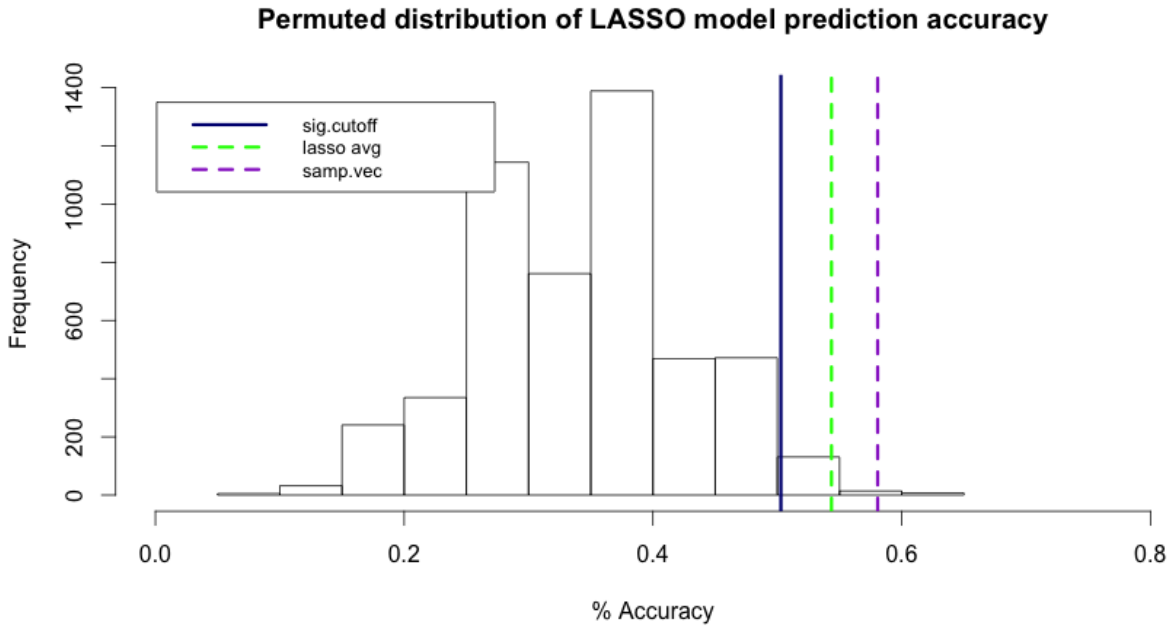
[1] The latter weighting schema seems more appropriate to employ given the dataset, as tf-idf places more emphasis on the terms that are specifically representative for each document (and therefore should be more likely to indicate condition membership). Even so, the two weighting schema really aren't that different.

prediction vector. With a distribution over the null we can then use standard null-hypothesis testing to assess whether our model's accuracy performs significantly better than chance[2].

Using the `boot` package, we ran 5000 permutations to generate a distribution of model accuracy under the null hypothesis, ie. that the model performs no better than random chance. There was in fact a large spread around the hypothetical 33% mark, as shown in the graph below. We determined whether our average accuracy rate was significantly better than chance using two different methods:

1) using the null distribution's mean and standard deviation to determine the 95% cutoff point (ie. z-score of 1.96),

2) using `pnorm()` to determine the quantile of our average accuracy score. Both methods indicated we could safely reject the null hypothesis that our model performed no better than chance (eg. using `pnorm()`, $p = 0.007$). In the graph below, markers indicate the significance cutoff (navy), the LASSO model average accuracy (green), and the accuracy average for the specific prediction vector used in the permutation (purple, see footnote 2 for more).



[2] Seeing as our test statistic (ie. 54%) was derived from an average across many model fittings, we weren't sure what to use as an appropriate vector of predictions - averaging predictions across the 100 fittings doesn't work as the response variable is made of discrete categories. As such, we simply chose to use the vector produced by the last model fit, as we figured it's as good a vector as any. The only other solution we considered was to run a permutation as part of each iteration of the model fitting, and then average the resulting p-values generated from NHST. We decided that might be overkill, so we chose the simpler solution.

Discussion

We find it extremely promising that a simple logistic regression was able to use a relatively crude measure of term frequency to accurately classify documents by the experimental condition to which their authors had been assigned. This type of analysis lends a degree of objectivity to qualitative findings that is frequently lacking in behavioral research. Further efforts should attempt to determine more specific understandings of when this sort of approach can yield fruitful results. Under conditions where writings could reasonably be expected to reflect group membership - such as the GOP data, where subjects were specifically asked to group-identify - this means of textual analysis is almost certainly appropriate. We were somewhat surprised to find that it can still be applied even to experimental designs which do not elicit specific social or political group identification, as in the Innovation Study data.

We would still like to find a way to link the community detection analysis originally performed with the GOP data to a prediction paradigm, although we were unable to do so within the scope of this project. One idea is to use directed edges (ie. term pairings), rather than nodes (ie. individual terms), as the elements in a feature set, and then evaluate the predictive capacity of each term pairing in the same logistic regression model. If network edges are also predictive of response class, and assuming edges are uniquely related to communities in a network, we could then examine whether the edges that are most related to a given response class are found more frequently in one community than in others. (This would require being able to examine regression coefficients for each predictor in relation to their classification of a given response class. The `glmnet predict()` object does have a `coef()` method, but we're not sure exactly how to use it toward this end.) As it is, the community detection of language sub-networks in a corpus can only really provide anecdotal classification; we believe this method can be significantly strengthened by developing it further to include a true predictive component.

Helpful links:
tm docs: http://cran.r-project.org/web/packages/tm/tm.pdf
glmnet docs: http://cran.r-project.org/web/packages/glmnet/glmnet.pdf
glmnet guide: http://www.stanford.edu/~hastie/glmnet/glmnet_alpha.html
matrix docs: http://cran.r-project.org/web/packages/Matrix/Matrix.pdf
pnorm: http://stat.ethz.ch/R-manual/R-devel/library/stats/html/Normal.html