STA 242: Assignment 4

Scott Frost and Tom Shapland

March 18, 2011

In this assignment we manipulated text data using regular expressions in order to import text data into R. Once the text data was in R, we explored features of the data using regular expressions and network plots. Our work focused on the Enron email data set.

1. Description of problems

First, we will describe the problems we encountered as we designed the code and our strategies for solving the problems.

We wanted to extract the header information (sender, recipient, date, and subject) and the body of the email. While it was easy to identify the lines containing this information using regular expressions, it was difficult to extract the information associated with these lines. We were advised to use the read.dcf() function, which befuddled Tom. Scott figured out how to use the read.dcf() command, but Tom decided to extract the data in a different manner. We will describe how Scott used the read.dcf() command and the alternative method that Tom used.

In Scott's method, we got around this problem doing x.

In Tom's method, we used regular expressions to identify lines in the data starting with the string “From: ”, which contained the sender information. In regular emails, the sender line is preceded by the date line, and followed by the recipient line and the subject line. Once we identified the sender lines, we indentified the date, recipient, and subject lines by their position relative to the sender line. The contents of the sender, date, recipient, and subject lines were extracted. Tom’s method ignored the email body.

In some cases the emails were malformed. In other words, the recipient line did not follow the sender line. If a line that follows a sender line does not start with “To: “, then the line was identified as coming from a malformed email. Malformed emails were culled. We may have lost important information in our culled emails, but hopefully we preserved enough material to indict Chairman and CEO Kenneth Lay before he suffers heart failure. Using Tom’s method, the total number of emails was 447,380 and the number of malformed emails was 144,050, i.e., nearly 1/3 of all the emails were malformed. Using Scott’s method, the total number of emails was X and the number of malformed emails was Y.

It was difficult to extract the header information and email bodies without running into memory problems. There were 27,390,842 lines in the Enron email data set. Initially Scott's method required reading in the whole data set in order to extract the header information and the email body. R was unable to read in 27 million lines.

Scott got around this problem doing x.

Tom also ran into memory problems with his method. Tom resolved this problem by reading 10,000 line segments of the data set into R, extracting the header information, outputting the header information in a file, then repeating the loop.

Only a small portion of our time was spent developing usable code. For most of our work in this assignment we did not use version control because we only had a few lines of quasi-functional code. Once we had a small amount of functional code, we used version control.

2. Interesting features of the Enron dataset

We will next describe interesting features of the Enron data set. We created 2 network maps. The first network map describes the first 10,000 emails in the data set. The second network map describes the email communications involving Ken Lay.

For the first network map we originally had hoped to produce a network map for the entire data set, but our machines crashed during the computation of the very large network plot. We instead used a subset of the email data consisting of the first 10,000 lines. Because Tom was unable to install Rgraphviz on any machine he touched, we decided to use the igraph package to create the network plot. The igraph package is available at the CRAN repository. The igraph package is subject to the requirements of packages on the CRAN repository, so it readily installs on most machines. The igraph package doesn’t seem to have functionality for weighting the edges, so we were unable to perform this sort of analysis.

We used a force-based algorithm, the Fruchterman*-*Reingold algorithm, for creating the layout of the network plot. Force-based algorithms make the nodes mutually repulsive, while the edges counteract the repulsive forces between the nodes. The plot has the aesthetically-pleasing characteristic of nearly equal spacing between the nodes in a network. If there are two groups of nodes on the same plot, i.e., there are two sets of mutually-exclusive nodes, then the two independent networks separate.

In the plot of the network of the first 10,000 emails of the data, we see 2 groups containing many nodes. We also see many more small groups. The large groups describe the email network of 2 individuals at Enron. The center of each large network is an Enron employee, who sent many emails to many different people. The small groups were likely created from email addresses that were included in the list of email recipients. In order to eliminate the small groups, we would have to make sure each email had only one sender and one recipient, which is unreasonable.

The plot of Kenneth Lay’s network included 4000 emails. …

We also explored the email data using regular expressions. We search Ken Lay’s emails for words such as “fraud” and “greed”, and text strings such as “nuts to the employee penchants!”.

3. Interesting features of the R-mailing list

Because we encountered so much difficulty in processing the Enron email data set, we did not have time to explore the R-mailing list.