Text Mining in R

Clay Ford, StatLab

March 5, 2014

What is Text Mining?

My informal definition: programming a computer to read, summarize and make decisions about text.
Two well-known examples:

1. spam filter
2. search engine

Basic idea: turn text into numbers and apply analytical algorithms.

Text Mining steps (usually)

1. Import text
2. Organize and structure text for access (create a corpus)
3. Transform text into something we can analyze (create a term-document matrix)
4. Do the analysis

On the agenda

Work examples in three practice areas:

1. classification - classify documents into known categories (supervised learning)
2. concept extraction - determine meaning of text (sentiment analysis)
3. clustering - group similar documents into clusters (unsupervised learning)

Why do Text Mining in R?

Because we can.

- R can manipulate text and files.
- R has the statistical functionality for analyses.
- Makes sense to leverage our knowledge of R (if you're already an R user).

Why do Text Mining in R?

Because we can.

- R can manipulate text and files.
- R has the statistical functionality for analyses.
- Makes sense to leverage our knowledge of R (if you're already an R user).

Should we do text mining in R?
Debatable. Perl and Python are certainly better at the text processing part. Memory issues can be a problem for R. Best for small-to-medium sized projects.

Not saying R is the ideal platform for text mining, just showing how it can be done.

Text Mining Terms

Good words to know:

- Corpus: database for holding text documents
- Term-Document Matrix: matrix displaying frequencies of words (rows) occuring in documents (columns)
- Stop Words: common words that do not contribute information
- Stemming: reduction of words to their simplest forms (roots)

Text Mining Terms

Good words to know:

- Corpus: database for holding text documents
- Term-Document Matrix: matrix displaying frequencies of words (rows) occuring in documents (columns)
- Stop Words: common words that do not contribute information
- Stemming: reduction of words to their simplest forms (roots)

Let's go to R...

Classification of Tweets

Background: The UVa library collected over 52,000 tweets during the Teresa Sullivan saga. Turns out many of the tweets were not related to the events surrounding Teresa Sullivan. Would like to programmatically classify tweets as relevant or not.

Classification of Tweets

Clay Ford, StatLab

Background: The UVa library collected over 52,000 tweets during the Teresa Sullivan saga. Turns out many of the tweets were not related to the events surrounding Teresa Sullivan. Would like to programmatically classify tweets as relevant or not.

About the text: Tweets stored as JSON files (JavaScript Object Notation). A way to store and transport text in a structured way. R has packages for importing JSON files.

Classification of Tweets

Background: The UVa library collected over 52,000 tweets during the Teresa Sullivan saga. Turns out many of the tweets were not related to the events surrounding Teresa Sullivan. Would like to programmatically classify tweets as relevant or not.

About the text: Tweets stored as JSON files (JavaScript Object Notation). A way to store and transport text in a structured way. R has packages for importing JSON files.

Strategy: Select a subset of tweets and classify manually. Then use this subset to train a model that will do it for the unclassified tweets. Example of supervised learning. We'll only use 1000 of the tweets.

Classification of Tweets - cont'd

About the analysis: logistic regression via lasso regularization.

Classification of Tweets - cont'd

About the analysis: logistic regression via lasso regularization.

- The lasso uses all predictors and regularizes, or shrinks, unimportant coefficients to 0. A form of model selection. Can also accommodate models where $p > n$.
- Includes a tuning parameter, λ. When $\lambda = 0$, lasso gives least squares fit. When λ is sufficiently large, all coefficients are 0. We must select a good value of λ. We use cross-validation to help with this.
- 10-fold Cross-Validation: split data into 10 sets. Use 9 of the sets to build a model and the remaining set to validate the model. Compare the classifications of the remaining set to its true classifications. Repeat for 9 other sets and find the average misclassification rate. Do this process for multiple values of λ and choose λ that yields lowest misclassification rate.

Sentiment Analysis of Steve Jobs Article

Background: *The New York Times* published a long article on Steve Jobs in Oct 2013. Would like to programmatically review user comments to get a feel for the sentiment expressed.

Sentiment Analysis of Steve Jobs Article

Background: *The New York Times* published a long article on Steve Jobs in Oct 2013. Would like to programmatically review user comments to get a feel for the sentiment expressed.
About the text: User comments in JSON format extracted using the New York Times API.

Sentiment Analysis of Steve Jobs Article

Background: *The New York Times* published a long article on Steve Jobs in Oct 2013. Would like to programmatically review user comments to get a feel for the sentiment expressed.
About the text: User comments in JSON format extracted using the New York Times API.
Strategy: Implement a naïve algorithm for scoring sentiment. Compare words in the comments to two opinion lexicons: one positive and one negative. Count the number of positive and negative matches and take the difference. Score = sum(positive) - sum(negative)

Clustering of Amazon Reviews

Background: I'm interested in the FitBit activity tracker. It has mostly positive reviews on Amazon but a sizable number of negative reviews. I'd like to know more about the negative reviews. Do they fall into any particular groups? Are there reoccurring reasons for negative reviews?

Clustering of Amazon Reviews

Background: I'm interested in the FitBit activity tracker. It has mostly positive reviews on Amazon but a sizable number of negative reviews. I'd like to know more about the negative reviews. Do they fall into any particular groups? Are there reoccurring reasons for negative reviews?

About the text: On the Amazon web site. Needs to be "scraped". In other words, copied off web site and cleaned up.

Clustering of Amazon Reviews

Background: I'm interested in the FitBit activity tracker. It has mostly positive reviews on Amazon but a sizable number of negative reviews. I'd like to know more about the negative reviews. Do they fall into any particular groups? Are there reoccurring reasons for negative reviews?
About the text: On the Amazon web site. Needs to be "scraped". In other words, copied off web site and cleaned up.
Strategy: Use a clustering algorithm to place the negative reviews into groups. This is unsupervised learning. I can't train a model to do this.

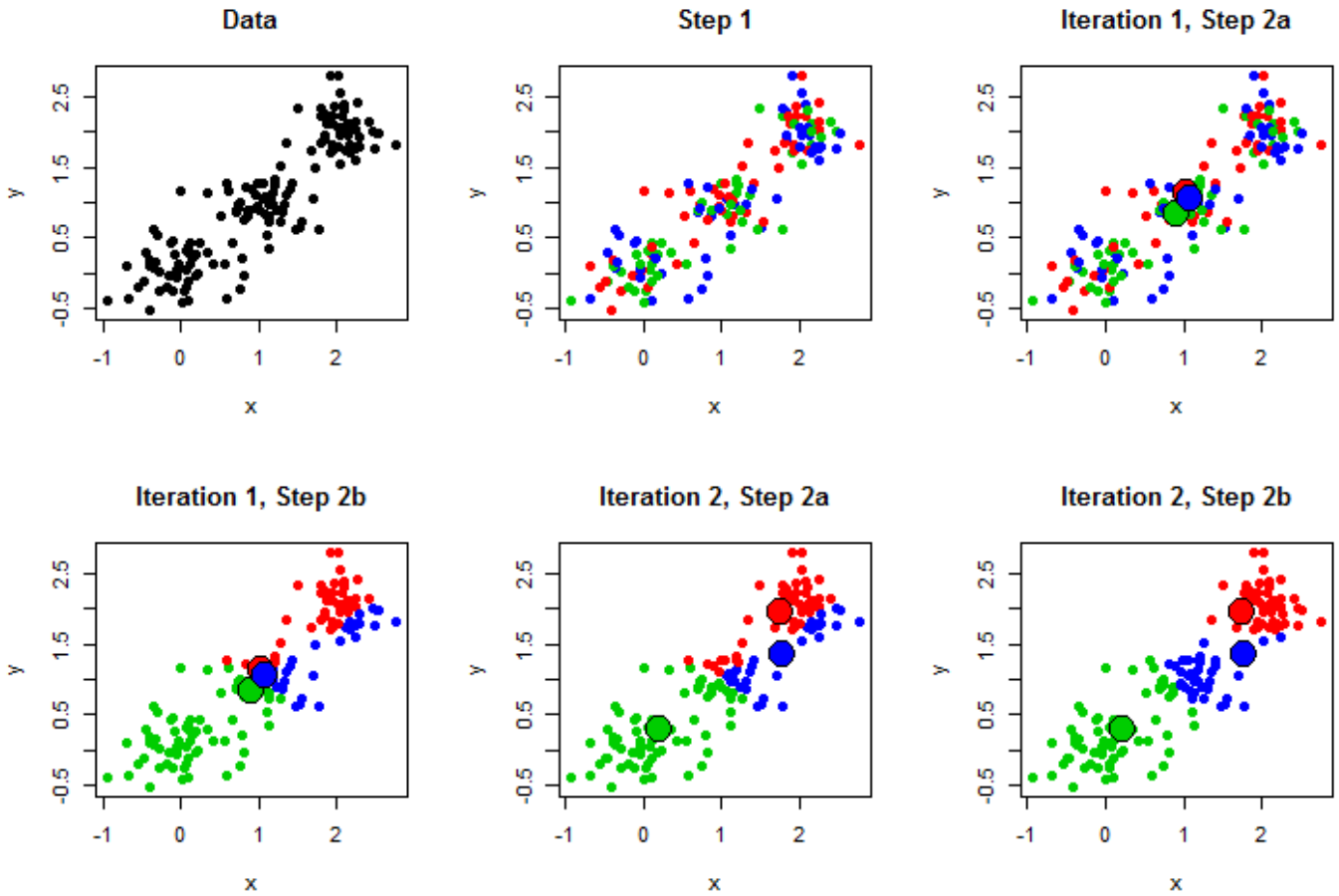
Clustering of Amazon Reviews - cont'd

About the analysis: K-means clustering: specify number of clusters (or groups) in advance, then assign each review to one of the groups using the k-means algorithm.
The k-means algorithm:

1. randomly assign each review to a group
2. iterate through the following until group assignments stop changing
    1. for each group, calculate the centroid (the vector of predictor means for the observations in each group)
    2. assign each observation to the cluster whose centroid is closest
3. Repeat steps 1 and 2 multiple times from different random assignments and select best "solution" (groups with smallest within-cluster variation).

Clustering of Amazon Reviews - cont'd

Example of k-means algorithm at work in 2D:



References

- Feinerer, et al. (2008). "Text Mining Infrastructure in R." *Journal of Statistical Software*, Vol. 25, Issue 5, Mar 2008.
- Feinerer (2008). "An Introduction to Text Mining in R." *R News*, Volume 8/2, Oct 2008, 19-22.
- James, et al. *An Introduction to Statistical Learning with Applications in R*, Springer, 2013.
- Conway and White, *Machine Learning for Hackers*, O'Reilly, 2012.
- Miner, et al. *Practical Text Mining*, Elsevier, 2012.