Text Classification in R

Michele Claibourn

March 2, 2017

Approaches for Working with Text

Michele Claibourn

Introduction

Supervised (Machine) Learning

Naive Bayes

Logit with Regularization

Assesing Models

SVMs and more

Grimmer and Stewart 2013

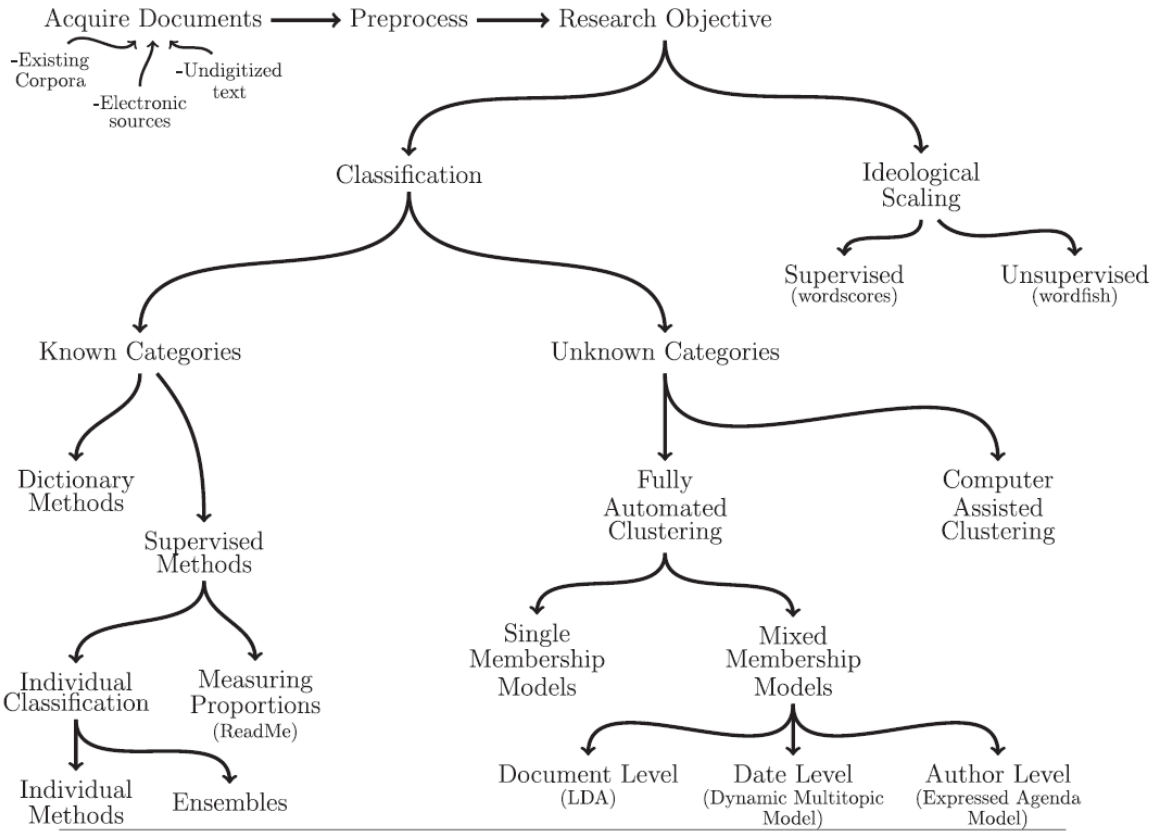


Fig. 1 An overview of text as data methods.

Approaches for Working with Text

O'Connor, Bamman, and Smith 2011

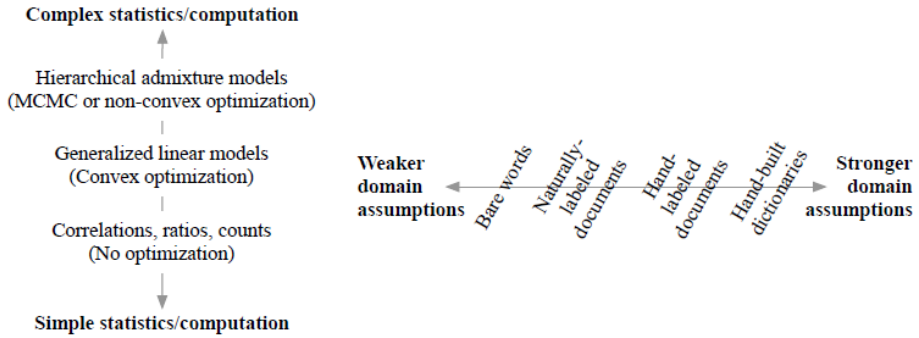


Figure 1: Schematic of model complexity versus domain assumptions for various computational text analysis methods. Statistical models are listed with their respective inference/training algorithms; computational expense increases with model expressiveness.

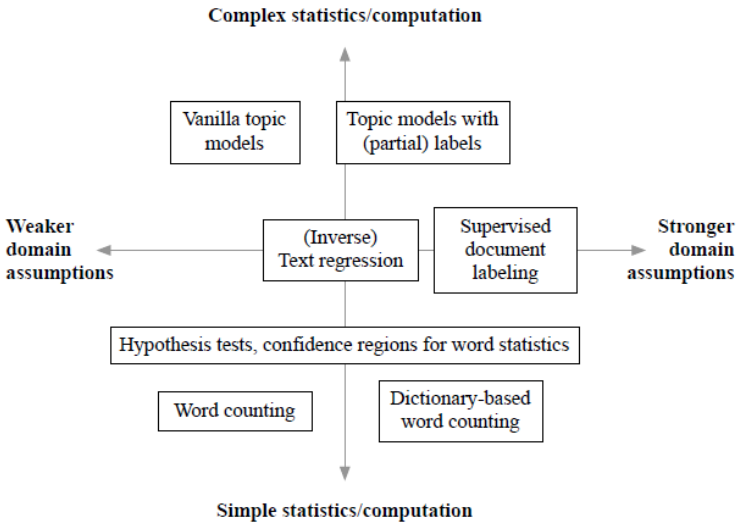


Figure 2: Typical methods used in computational text analysis. Compare to Table 1 in [27].

Classification

Goal: place text into a pre-defined categories, e.g.,

- Topics or issues, e.g., policy areas for legislation
- Source or authors, e.g., stylometry
- Spam or other filters (e.g., sexually explicit content)
- Sentiment or ratings, e.g., negative/positive

Classification

Goal: place text into a pre-defined categories, e.g.,

- Topics or issues, e.g., policy areas for legislation
- Source or authors, e.g., stylometry
- Spam or other filters (e.g., sexually explicit content)
- Sentiment or ratings, e.g., negative/positive

Approaches:

- Dictionaries: pre-identified words that associate with classes are counted/weighted
- **Supervised classification**: statistical models identify separating words

Some Notation

The goal is to classify every document into one category:

- i documents, $(i = 1, \ldots, N)$ with j features: $x_i = (x_{1i}, x_{2i}, \ldots, x_{ji})$
- k categories, $(k = 1, \ldots, K)$ such that $\{C_1, C_2, \ldots, C_K\}$
- Labeled documents, $Y = (Y_1, Y_2, \ldots, Y_{N_{train}})$ such that $Y_i \in \{C_1, C_2, \ldots, C_K\}$

Generally, we'll define some objective function $y = f(X, \theta)$ where $\theta = (\theta_1, \theta_2, \ldots, \theta_N)$ are feature weights.

Supervised Classification

- Human coders classify a subset, N_{train} documents, into predetermined categories (or text is conveniently pre-coded).
  - Need clear categories, simple coding rules, and trained coders. This is hard!
  - Use multiple coders for (at least some) documents to test inter-coder reliability (e.g., Krippendorf's α, Cohen's κ).
  - Produce a labeled set for training, a labeled set for validation. How many?
- Labeled documents are used to train a model. Optimize with respect to θ to "learn" the weights.
- Model is validated against another hand-labeled subset, N_{test}. Obtain predicted fit for new data $f(X_i, \hat{\theta})$ and compare to known categories.
- Chosen model is applied to unlabeled cases.

Naive Bayes Classifier

A simple application of Bayes' rule; often surprisingly useful.
For each document i, we want to infer the most likely category C_k, based on features of the document x_i

$$C_{Max} = \arg\max_k \; p(C_k|x_i)$$

Naive Bayes Classifier

A simple application of Bayes' rule; often surprisingly useful. For each document i, we want to infer the most likely category C_k, based on features of the document x_i

$$C_{Max} = \arg\max_k \; p(C_k|x_i)$$

Use Bayes' rule to estimate $p(C_k|x_i)$.

$$\begin{aligned} p(C_k|x_i) &= \frac{p(C_k, x_i)}{p(x_i)} \\ &= \frac{p(C_k)p(x_i|C_k)}{p(x_i)} \end{aligned}$$

Naive Bayes Classifier

A simple application of Bayes' rule; often surprisingly useful. For each document i, we want to infer the most likely category C_k, based on features of the document x_i

$$C_{Max} = \arg\max_k \; p(C_k|x_i)$$

Use Bayes' rule to estimate $p(C_k|x_i)$.

$$\begin{aligned} p(C_k|x_i) &= \frac{p(C_k, x_i)}{p(x_i)} \\ &= \frac{p(C_k)p(x_i|C_k)}{p(x_i)} \end{aligned}$$

Two probabilities to estimate

- $P(C_k) = \frac{\#\text{ docs in } k}{\#\text{ docs in } N_{train}}$
- $P(x_i|C_k)$

The second is complicated unless we make the simplifying assumption: features, x_i, are independent. Ergo... "naive"

Naive Bayes Classifier

$$p(x_i|C_k) = \prod_{j=1}^{J} p(x_{ij}|C_k)$$

This is what we're maximizing. Classification is made using maximum a posteriori (MAP) decision rule.

Naive Bayes Classifier

$$p(x_i|C_k) = \prod_{j=1}^{J} p(x_{ij}|C_k)$$

This is what we're maximizing. Classification is made using maximum a posteriori (MAP) decision rule.
For any given feature of a document z, we have

$$P(x_{im} = z|C_k) = \frac{\#(x_{im} = w \text{ and } C = C_k)}{\#(C = C_k)}$$

What happens when a word-category combination doesn't appear in the training data? This is zero!

Naive Bayes Classifier

$$p(x_i|C_k) = \prod_{j=1}^{J} p(x_{ij}|C_k)$$

This is what we're maximizing. Classification is made using maximum a posteriori (MAP) decision rule.
For any given feature of a document z, we have

$$P(x_{im} = z|C_k) = \frac{\#(x_{im} = w \text{ and } C = C_k)}{\#(C = C_k)}$$

What happens when a word-category combination doesn't appear in the training data? This is zero! To eliminate zeroes, smooth the estimate, e.g., Laplace smoothing, adds one to each count

$$P(x_{im} = z|C_k) = \frac{\#(x_{im} = w \text{ and } C = C_k) + 1}{\#(C = C_k)}$$

Logit with Lasso Regularization

Logistic regression

- Like standard linear regression, but for binary outcomes
- Rather than predict outcome, y, predict $p(y = 1)$
- Because probabilites are bounded, use a cumulative probability distribution – an S-shaped curve
- Select a threshold to map probabilities into binary outcome, e.g., $p > .5 = 1$ and $p < .5 = 0$

Logit with Lasso Regularization

Logistic regression

- Like standard linear regression, but for binary outcomes
- Rather than predict outcome, y, predict $p(y = 1)$
- Because probabilites are bounded, use a cumulative probability distribution – an S-shaped curve
- Select a threshold to map probabilities into binary outcome, e.g., $p > .5 = 1$ and $p < .5 = 0$

Regularization

- Lots of features, $p >> n$ problem; and features are highly correlated
- Leads to more variable estimates and overfitting
- Regularization constrains the coefficient estimates, shrinks them towards zero

Logit with Lasso Regularization

The Lasso

- Uses all predictors, shrinks less important ones to 0
- Adds a tuning paramter, λ, as penalty to model complexity
- When $\lambda = 0$ Lasso produces least-squares/MLE fit; as $\lambda \rightarrow \infty$ all coefficients approach zero

Logit with Lasso Regularization

The Lasso

- Uses all predictors, shrinks less important ones to 0
- Adds a tuning paramter, λ, as penalty to model complexity
- When $\lambda = 0$ Lasso produces least-squares/MLE fit; as $\lambda \rightarrow \infty$ all coefficients approach zero

N-fold cross-validation to choose λ

- Split data into N sets
- Use $N - 1$ sets to build model and omitted set to validate
- Repeat for all N sets
- Average the misclassification rate

Do this for a range of λ and choose λ with lowest average misclassification

The Confusion Matrix, and other metrics

	Predicted Class	
True Class	Positive	Negative
Positive	True Positive	False Negative
Negative	False Positive	True Negative

$$Accuracy = \frac{TruePos + TrueNeg}{TruePos + TrueNeg + FalsePos + FalseNeg}$$

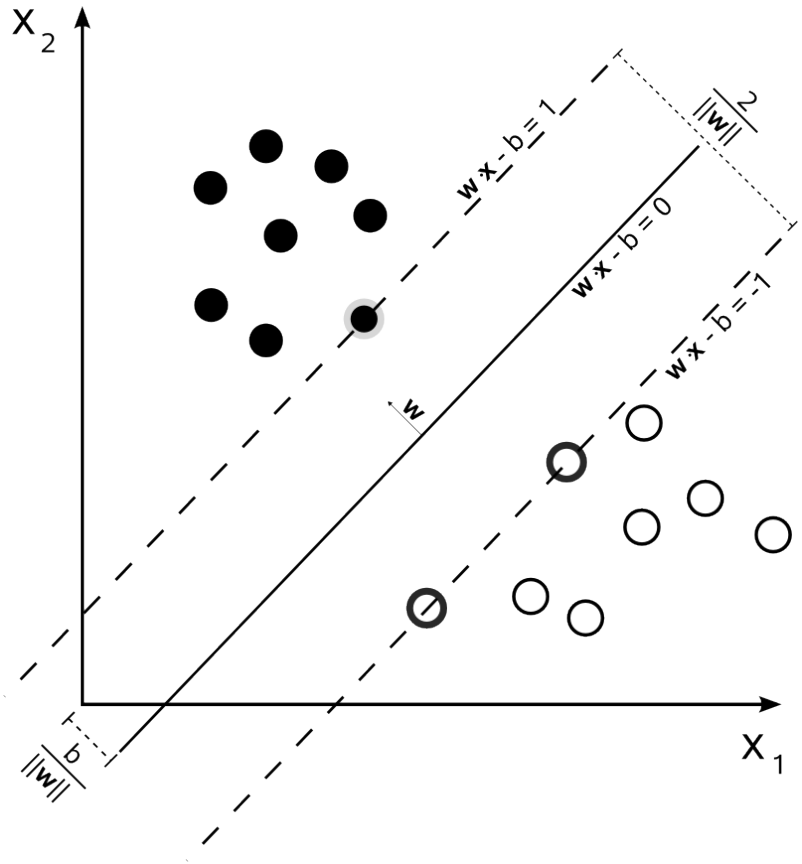
$$Precision_{pos} = \frac{TruePos}{TruePos + FalsePos}$$

$$Recall_{pos} = \frac{TruePos}{TruePos + FalseNeg}$$

$$F1_{pos} = 2 * \frac{Precision_{pos} \times Recall_{pos}}{Precision_{pos} + Recall_{pos}}$$

Support Vector Machines

Seeks the optimal separating hyperplane between classes, maximizing the margin between the closest points (the support vectors)



Support Vector Machines

- The C parameter adds a penalty for misclassification, balancing misclassification and maximization of the hyperplane. We minimize

$$\|\omega\|^2 + C\sum_{i=1}^{n}\xi$$

- The kernal fuction extends the algorithm to patterns that are not linearly separable by mapping the original data into a higher dimensional space, e.g.,

$$K(x, y) = (x \cdot y + 1)^p \qquad \textit{polynomial}$$

$$K(x, y) = exp\left(-\frac{\|x - y\|^2}{2\sigma^2}\right) \qquad \textit{radial}$$

$$K(x, y) = tanh(\kappa x \cdot y - \delta) \qquad \textit{sigmoid}$$

Yet more Algorithms

`RTextTools` implements nine algorithms (there are many more possibilities), including

- Maximum entropy (MAXENT, from `maxent`)
- Classification trees (TREE, from `tree`)
- Random forests (RF, from `randomForest`)
- Bootstrap aggregation (BAGGING, from `ipred`)
- Linear discriminant analysis (SLDA¡ from `ipred`)
- Logit boost classifier (BOOSTING, from `caTools`)
- Neural net (NNET, from `nnet`)

And ensemble approaches: bagging, boosting, stacking...