Introduction to NLP

Practical Issues in Text Classification
Pitfall: Overfitting

What happens when your model learns your training data a little too well?
Development Test Sets and Cross-validation

- **Metric:** P/R/F1 or Accuracy
- **Unseen test set**
  - avoid overfitting (‘tuning to the test set’)
  - more conservative estimate of performance
- **Cross-validation over multiple splits**
  - Handle sampling errors from different datasets
  - Pool results over each split
  - Compute pooled dev set performance
Underflow Prevention: log space

- Multiplying lots of probabilities can result in floating-point underflow.
- Since \( \log(xy) = \log(x) + \log(y) \)
  - Better to sum logs of probabilities instead of multiplying probabilities.
- Class with highest un-normalized log probability score is still most probable.

\[
c_{NB} = \arg\max_{c_j \in C} \log P(c_j) + \sum_{i \in \text{positions}} \log P(x_i | c_j)
\]

- Model is now just max of sum of weights
No labeled data: Manual Rules

If (wheat or grain) and not (whole or bread) then
Categorize as grain

• Need careful crafting
  – Human tuning on development data
  – Time-consuming: 2 days per class
Very little data?

• Use Naïve Bayes
  – Naïve Bayes is a “high-bias” algorithm (Ng and Jordan 2002 NIPS)

• Get more labeled data
  – Find clever ways to get humans to label data for you

• Try semi-supervised training methods:
  – Bootstrapping, EM over unlabeled documents, ...
A reasonable amount of data?

• Perfect for all the clever classifiers
  – SVM
  – Regularized Logistic Regression

• You can even use user-interpretable decision trees
  – Users like to hack
  – Management likes quick fixes
A huge amount of data?

• Can achieve high accuracy!
• At a cost:
  – SVMs (train time) or kNN (test time) can be too slow
  – Regularized logistic regression can be somewhat better
• So Naïve Bayes can come back into its own again!
• Or you can play with deep learning...
NLP