

Shrinkage Techniques for Hierarchical Text Classification*

Mohsen Azarbayejani

May 5, 2005

CSE254

*based on “Improving Text Classification by Shrinkage in a Hierarchy of Classes”, by A. McCallum *et al*, 15th *International Conference on Machine Learning (ICML-98)*

Outline

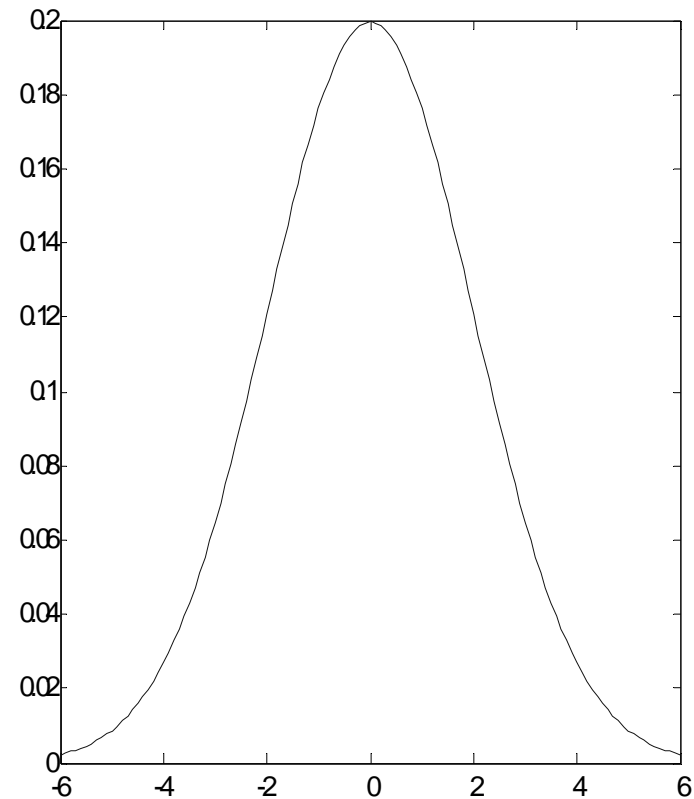
- Shrinkage in statistics
- The task: learning to classify documents
- The Bayesian approach
- Shrinkage applied to document classification
- Experimental design
- Experimental results

Shrinkage in Statistics

- Given several different estimates of the same quantity (for example, by different methods)...
- The estimates can be improved by scaling and shifting them all toward a common value, thereby *shrinking* the variance of each.

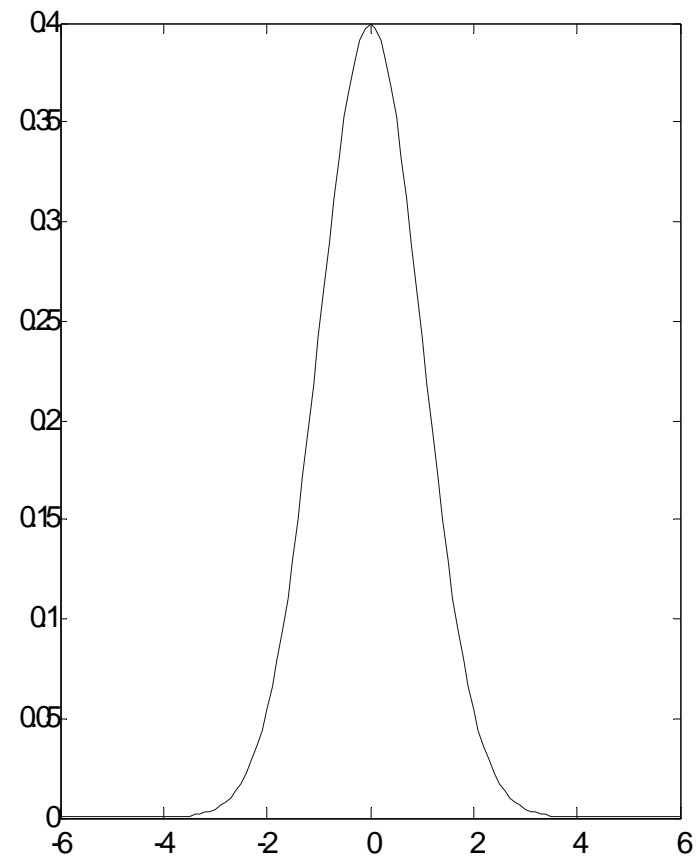
Example of Shrinkage

- Suppose several points are sampled from a Gaussian distribution.
- Each point is an estimate of the mean.



Example of Shrinkage

- Shrinkage decreases the variance.
- All points are now closer to the actual mean.

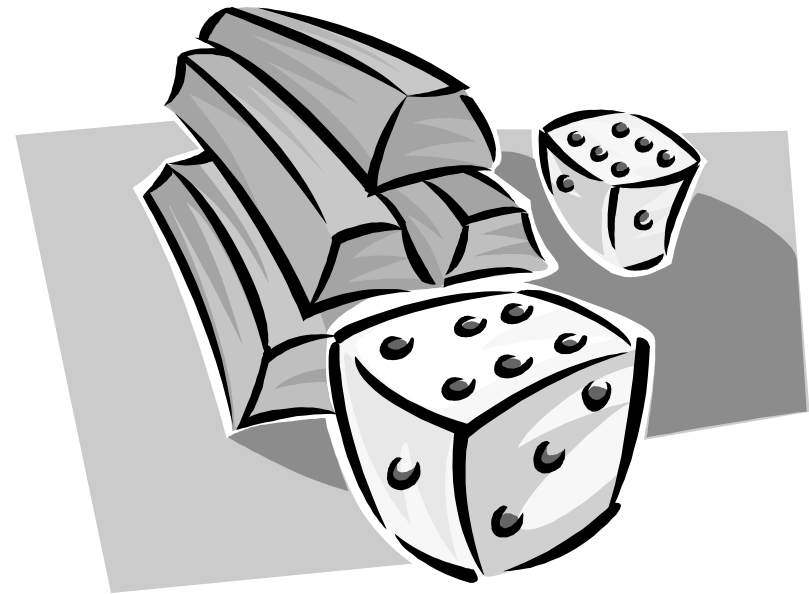


The Text Classification Task

- Given a text document, decide which of several pre-defined categories it belongs to.
- The count of each word gives the document a certain probability of belonging to each category.
- For example, the word *loss* correlates with Business and Sports more than with Entertainment.

The Multinomial Model

The multinomial distribution gives the probability of events like rolling N_1 ones, N_2 twos, ... N_6 sixes out of n total rolls of a die:



$$p(N_1, \dots, N_6) = \frac{n! \prod_{t=1}^6 p_t^{N_t}}{\prod_{t=1}^6 N_t!} \quad \text{where } n = \sum_{t=1}^6 N_t$$

The Multinomial Model Applied to Document Classification

Replacing counts of specific rolls of the die with word counts in a document yields:

$$p(N_1, \dots, N_m) = \frac{|d|! \prod_{t=1}^m [p(w_t)]^{N_t}}{\prod_{t=1}^m N_t!}$$

Word Types vs. Tokens

In these equations:

- m is the number of unique words (*types*) in the document.
- $|d|$ is the total number of words (*tokens*) in the document.
- N_t is the count (*tokens / type*) for word i .
- These quantities are related as follows:

$$|d| = \sum_{t=1}^m N_t$$

Representation of Documents

Each document d_i is represented as a vector of word counts, generated from a multinomial distribution.

$$p(d_i) = p(N_{i1}, \dots, N_{im}) = \frac{|d_i|! \prod_{t=1}^m [p(w_t)]^{N_{it}}}{\prod_{t=1}^m N_{it}!}$$

Parameters of Multinomial Document Model

$$p(d_i) = p(N_{i1}, \dots, N_{im}) = \frac{|d_i|! \prod_{t=1}^m [p(w_t)]^{N_{it}}}{\prod_{t=1}^m N_{it}!}$$

This model is parameterized by the probabilities of each word, $p(w_t)$.

Modeling Document Classes

- Documents that have similar word count vectors belong to the same category.
- Categories or classes are modeled as clusters of documents in word-count space.

Mixture Model of Classes

The overall probability of a document is a mixture of multinomial distributions of the probability of the document belonging to each class.

$$p(d) = \sum_j p(d | c_j) p(c_j)$$

Multinomial Document Model With Categories

$$p(d_i | c_j) = p(N_{i1}, \dots, N_{im} | c_j) = \frac{|d_i|! \prod_{t=1}^m [p(w_t | c_j)]^{N_{it}}}{\prod_{t=1}^m N_{it}!}$$

This model is parameterized by the probabilities of each word given each category, $p(w_t/c_j)$.

Naive Bayes Assumption

Ignoring the normalizing constants, this formula can be expressed in terms of the probability of each token in the document given the class:

$$p(d_i | c_j) \propto \prod_{t=1}^m \left[p(w_t | c_j) \right]^{N_{it}} = \prod_{k=1}^{|d_i|} p(w_{ik} | c_j)$$

This form of the equation demonstrates that the model makes the *naive Bayes* assumption, that the probabilities of each word are conditionally independent.

Bayesian Text Classification

$$p(c_j | d_i) = \frac{p(d_i | c_j) p(c_j)}{\sum_{r=1}^{|C|} p(d_i | c_r) p(c_r)}$$

- Given document d_i , find the category c_j , that it belongs to.
- Two parameters must be learned:
 - The probability that the category c_j generated document d_i .
 - The prior probability of category c_j .

Training the Bayesian Classifier

Bayes Rule takes the following form:

$$p(c_j | d_1, \dots, d_n) = \frac{p(d_1, \dots, d_n | c_j) p(c_j)}{\sum_{r=1}^{|\mathcal{C}|} p(d_1, \dots, d_n | c_r) p(c_r)}$$

Bayesian Learning and Text Classification

After training, the document can be categorized by using Bayes' Rule again:

$$p(c_j | d_i) = \frac{p(c_j) \prod_{k=1}^{|d_i|} p(w_{ik} | c_j)}{\sum_{r=1}^{|C|} p(c_r) \prod_{k=1}^{|d_i|} p(w_{ik} | c_r)}$$

Estimating the Class Priors

$$p(c_j) = \frac{\sum_{r=1}^{|D|} p(c_j | d_i)}{|D|}$$

where $p(c_j | d_i) \in \{0, 1\}$

The class prior is simply the number of training documents in the class divided by the total number of documents.

Estimating the “Word Given Class” Likelihood

$$\hat{\theta}_{jt} = p(w_t | c_j) = \frac{\sum_{i=1}^{|D|} N(w_t, d_i) p(c_j | d_i)}{\sum_{s=1}^{|V|} \sum_{i=1}^{|D|} N(w_s, d_i) p(c_j | d_i)}$$

The Maximum Likelihood estimate of the word | class likelihood is the number of *hits* (occurrences of the word in the document) for the given class, normalized by dividing by the total number of hits.

Zero Likelihood

$$\hat{\theta}_{jt} = p(w_t | c_j) = \frac{\sum_{i=1}^{|D|} N(w_t, d_i) p(c_j | d_i)}{\sum_{s=1}^{|V|} \sum_{i=1}^{|D|} N(w_s, d_i) p(c_j | d_i)}$$

If word w_t does not occur in class c_j during training, then:

$$\hat{\theta}_{jt} = p(w_t | c_j) = 0$$

Zero Likelihood

$$p(c_j | d_i) = \frac{p(c_j) \prod_{k=1}^{|d_i|} p(w_{ik} | c_j)}{\sum_{r=1}^{|C|} p(c_r) \prod_{k=1}^{|d_i|} p(w_{ik} | c_r)}$$

If a word occurs in a test document d_i that did not occur during training, then:

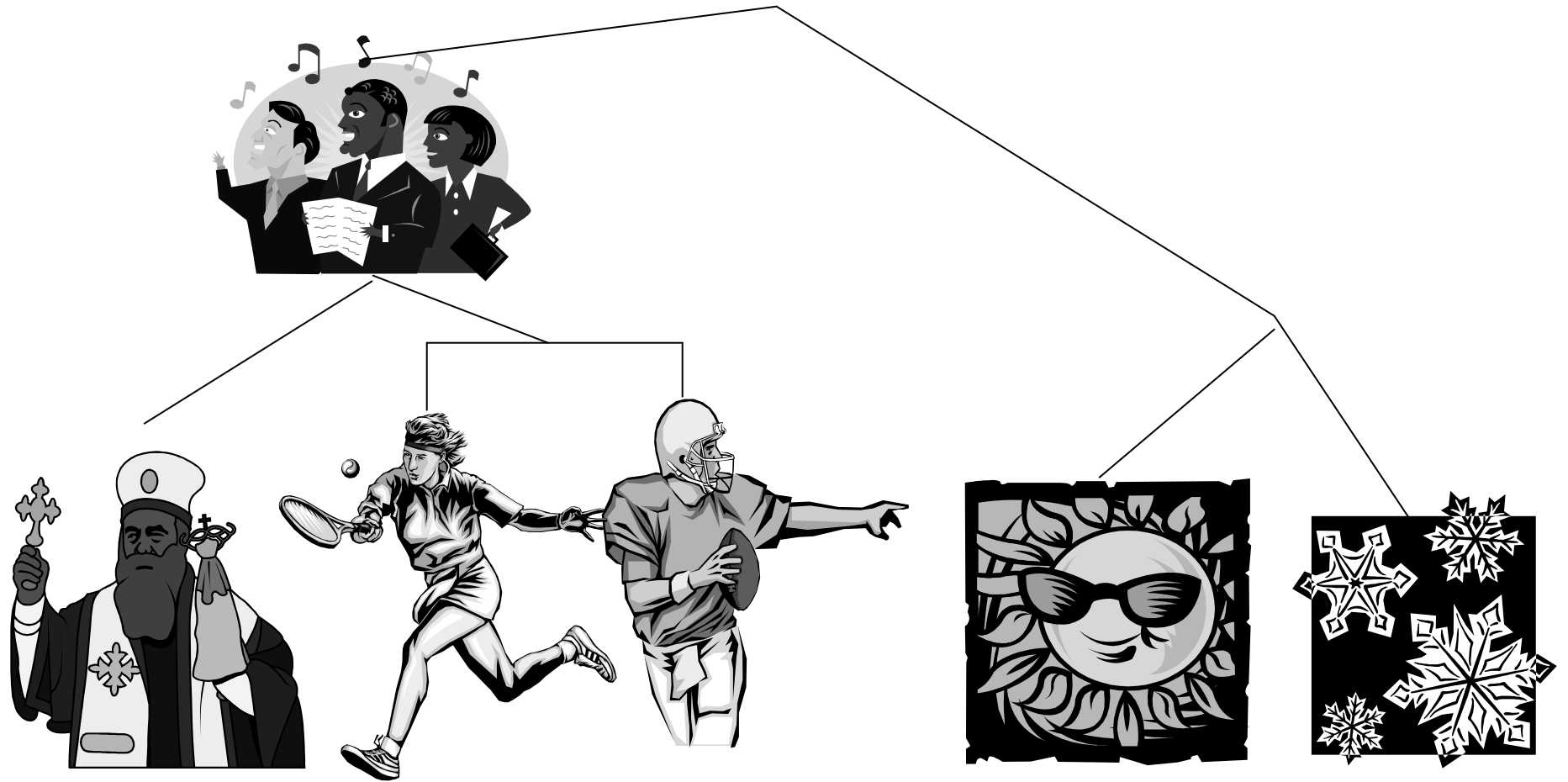
$$p(c_j | d_i) = 0$$

Smoothing the “Word Given Class” Likelihood

$$p(w_t | c_j) = \frac{1 + \sum_{i=1}^{|D|} N(w_t, d_i) p(c_j | d_i)}{|V| + \sum_{s=1}^{|V|} \sum_{i=1}^{|D|} N(w_s, d_i) p(c_j | d_i)}$$

To avoid zero values, the likelihood is *smoothed* by incrementing all counts by one occurrence.

A Topic Hierarchy



Topic Hierarchy

- Topics (or classes of documents) exist in a hierarchy.
- Each *leaf class* belongs to a super-class, which may belong to another super-class, and so on up to the root of the tree.
- There is a likelihood that each word indicates each class, but there is also a likelihood that each word indicates the super-classes of that class.

Shrinkage in the Hierarchy

Shrinkage improves θ_{jt} , the estimated probability of word w_t given class c_j , by bringing it closer to θ_{jt}^k for super-classes k , through linear combination:

$$\tilde{\theta}_{jt} = p(w_t | c_j) = \lambda_j^0 \hat{\theta}_{jt}^0 + \lambda_j^1 \hat{\theta}_{jt}^1 + \lambda_j^2 \hat{\theta}_{jt}^2 + \dots + \lambda_j^k \hat{\theta}_{jt}^k$$

where $\sum_k \lambda_j^k = 1$ for each class j .

Shrinkage with Smoothing

Smoothing is done by appending a “uniform prior” beyond the root of the class hierarchy.

$$p(w_t | c_j) = \lambda_j^0 p(w_t | c_j) + \dots \\ + \lambda_j^k p(w_t | c_j^k) + \lambda_j^{k+1} \frac{1}{|V|}$$

Shrinkage Algorithm

- Calculate initial parameter estimates using word counts (except on “hold-out set”).
- Initialize weights for the super-classes to any normalized value, such as $\lambda_j^i = 1/k$.
- Repeat until convergence:
 - Expectation step
 - Maximization step

Algorithm: Expectation Step

For each word w_t in the hold-out set of documents \mathbf{H}_j :

- » For each class from the leaf to the root:
 - > Calculate the “word given class” likelihood.
 - > Find the value of the vector λ corresponding to the likelihood values of the previous estimate.

$$\begin{aligned}\beta_j^i &= \sum_{w_t \in \mathcal{H}_j} P(\hat{\theta}_j^i \text{ was used to generate } w_t) \\ &= \sum_{w_t \in \mathcal{H}_j} \frac{\lambda_j^i \hat{\theta}_{jt}^i}{\sum_m \lambda_j^m \hat{\theta}_{jt}^m}\end{aligned}$$

Algorithm: Maximization Step

Calculate the optimal λ for each class that minimizes error across all words.

$$\lambda_j^i = \frac{\beta_j^i}{\sum_m \beta_j^m}$$

Experiments: Data Sets

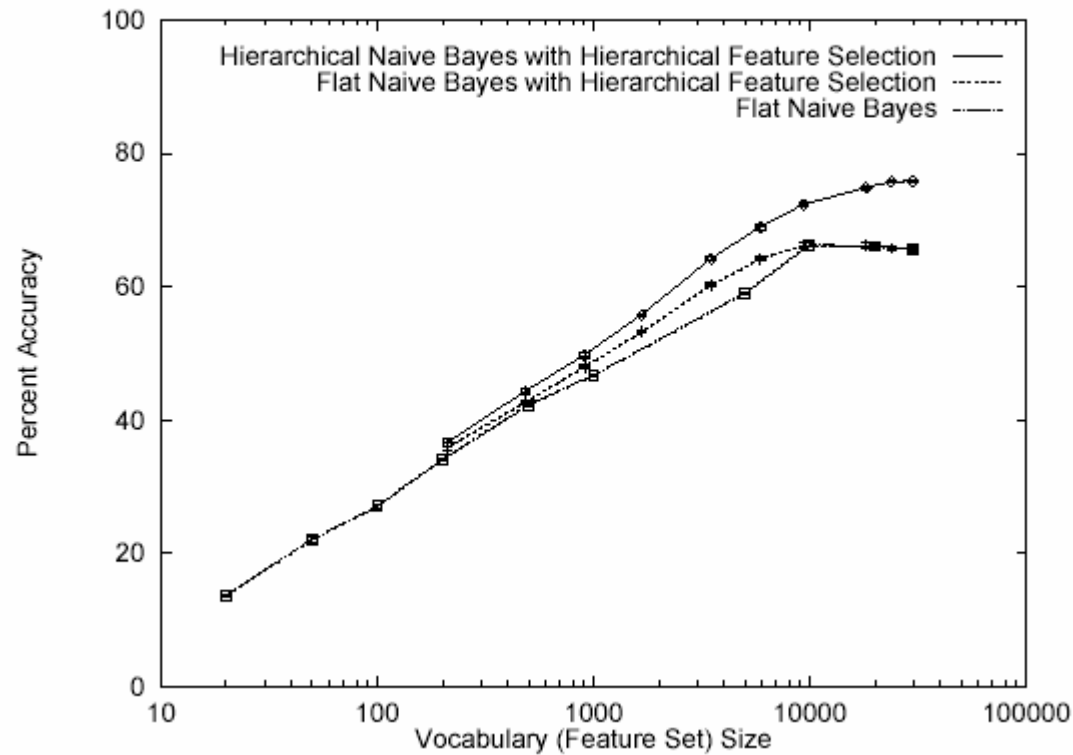
The shrinkage algorithm is tested on three hierarchical sets of documents:

- Company web pages, classified by industry sector.
- Usenet articles, classified into 20 newsgroup categories.
- Web pages from the Yahoo “science” and “health” hierarchies.

Experimental Design

- To ensure that probability estimates in the hierarchy are independent, the parameters of a super-class are determined only from the documents in “sister classes” of the original class, not the original class, itself.
- Hierarchical feature selection was used. This is a method of selecting features (words) with the highest mutual information between a class and its sub-classes.

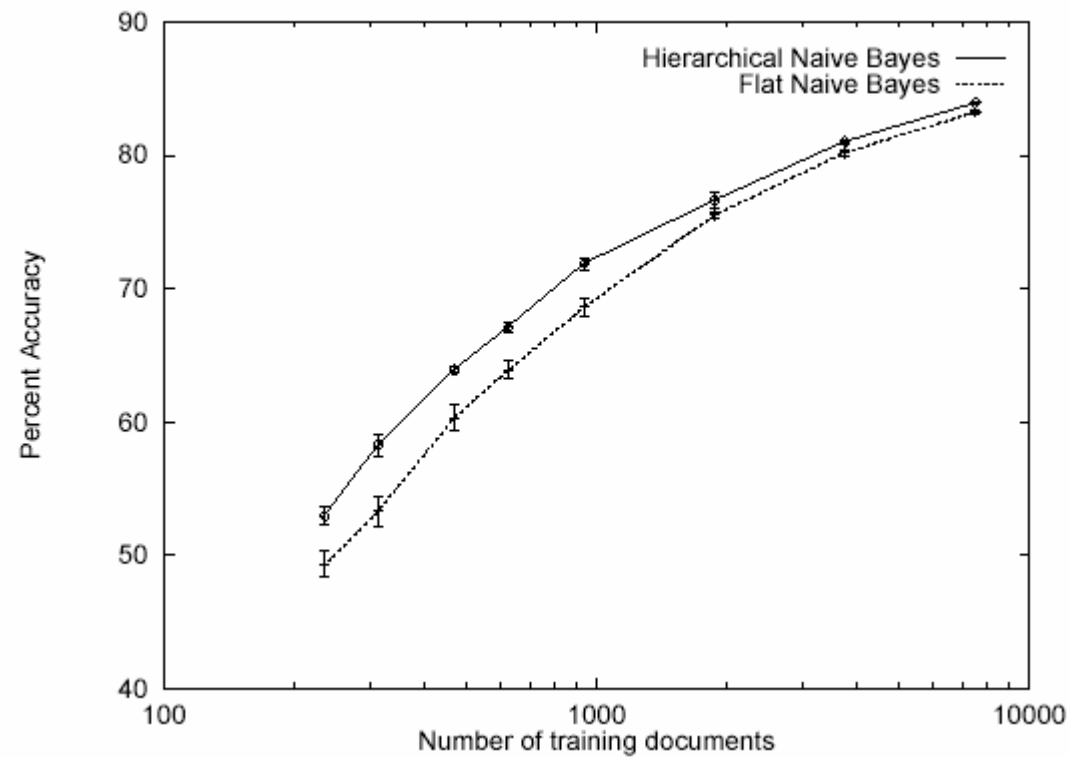
Results: Industry Sector



Results: Industry Sector

- The accuracy of all three classifiers improves with vocabulary size.
- Hierarchical feature selection improves the naive Bayes classifier around 5000 words.
- Shrinkage performs better than naive Bayes for all vocabulary sizes
- Shrinkage does not plateau at 10,000 words.

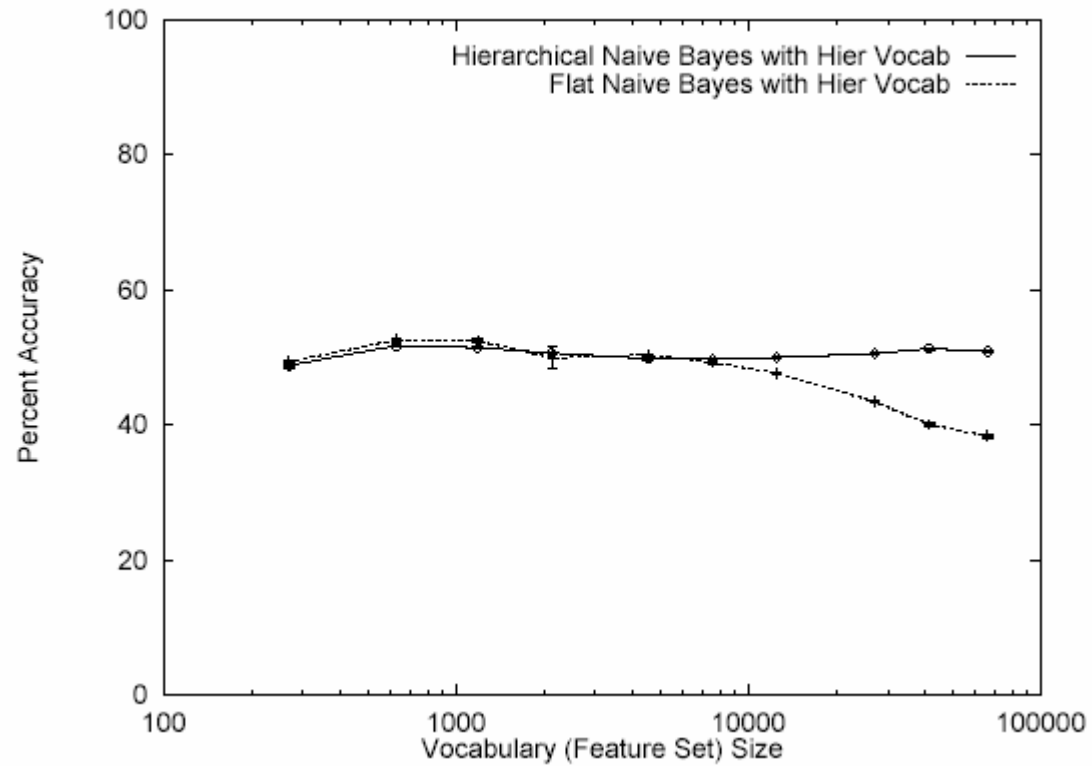
Results: Newsgroups



Results: Newsgroups

- In this experiment, the vocabulary size is held constant, while the size of the training set varies.
- Hierarchical feature selection does not improve either classifier and is not shown.
- Shrinkage performs better than naive Bayes, but by a smaller margin, due to lower fan-out.
- Shrinkage performs better than naive Bayes by a greater margin for smaller training sets.

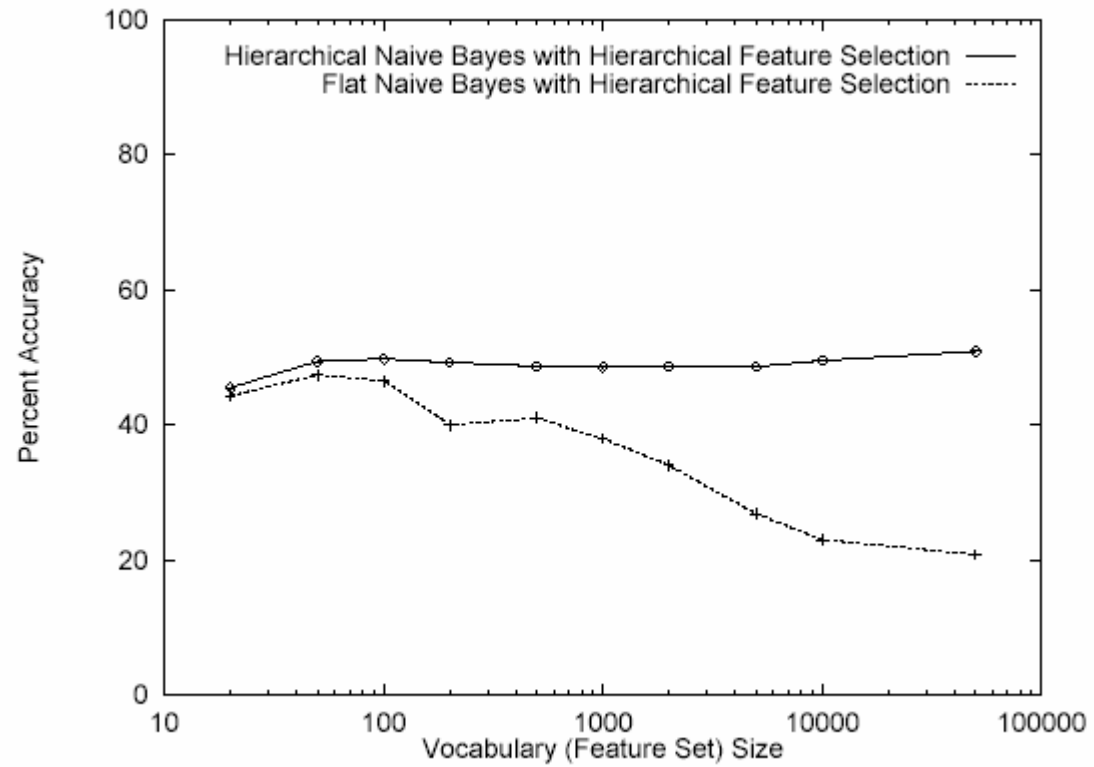
Results: Yahoo



Results: Yahoo

- Accuracy does not improve with vocabulary size for either classifier.
- Accuracy is actually reduced for naive Bayes above 10,000 words.
- The lack of improvement may be an artifact of the experimental setup, which can only handle hierarchies of depth two.

Results: Yahoo



Results: Yahoo

- The second graph shows accuracy averaged over classes instead of documents.
- Shrinkage shows a much greater performance improvement over naive Bayes.
- This is due to the nature of the Yahoo training set, in which the majority of documents were lumped into a few classes.

Conclusions

- Topic hierarchies can be leveraged to improve naive Bayes classifiers by using hierarchical feature selection.
- Classifiers can be further improved by using shrinkage.
- These results are particularly noticeable for small training sets or large vocabularies.


```
ERROR: undefined
OFFENDING COMMAND:

STACK:
```