Text as Data

Justin Grimmer

Associate Professor
Department of Political Science
Stanford University

November 6th, 2014
Supervised Learning

1) Task
   - Classify documents to pre existing categories
   - Measure the proportion of documents in each category

2) Objective function
   - Suppose we have $K$ categories.
   - Select $N_{\text{train}}$ document to hand-label, $Y_i = k$, $Y = (Y_1, Y_2, \ldots, Y_{N_{\text{train}}})$

   $$Y = f(X, \theta)$$

3) Optimization
   - Method specific: MLE, Bayesian, EM, ...
   - We learn $\hat{\theta}$

4) Validation
   - Obtain predicted fit for new data $f(X_i, \hat{\theta})$
   - Examine prediction performance $\Rightarrow$ compare classification to gold standard
Supervised Learning

Clustering and Topic Models:
- Models for discovery
  - Infer categories
  - Infer document assignment to categories
  - Pre-estimation: relatively little work
  - Post-estimation: extensive validation testing
Supervised Learning

Clustering and Topic Models:
- Models for discovery
  - Infer categories
  - Infer document assignment to categories
  - Pre-estimation: relatively little work
  - Post-estimation: extensive validation testing

Supervised Methods:
Supervised Learning

Clustering and Topic Models:
- Models for discovery
  - Infer categories
  - Infer document assignment to categories
  - Pre-estimation: relatively little work
  - Post-estimation: extensive validation testing

Supervised Methods:
- Models for categorizing texts
Supervised Learning

Clustering and Topic Models:
- Models for discovery
  - Infer categories
  - Infer document assignment to categories
  - Pre-estimation: relatively little work
  - Post-estimation: extensive validation testing

Supervised Methods:
- Models for categorizing texts
  - Know (develop) categories before hand
Supervised Learning

Clustering and Topic Models:
- Models for discovery
  - Infer categories
  - Infer document assignment to categories
  - Pre-estimation: relatively little work
  - Post-estimation: extensive validation testing

Supervised Methods:
- Models for categorizing texts
  - Know (develop) categories before hand
  - Hand coding: assign documents to categories
  - Infer: new document assignment to categories (distribution of documents to categories)
Supervised Learning

Clustering and Topic Models:
\- Models for discovery
  \- Infer categories
  \- Infer document assignment to categories
  \- \textbf{Pre-estimation}: relatively little work
  \- \textbf{Post-estimation}: extensive validation testing

Supervised Methods:
\- Models for categorizing texts
  \- \textbf{Pre-estimation}: extensive work constructing categories, building classifiers
  \- \textbf{Post-estimation}: relatively little work
Supervised Learning

Today:

- How to generate valid hand coding categories
- Assessing coder performance
- Assessing disagreement among coders
- Evidence coders perform well
- Supervised Learning Methods: Naive Bayes
- Assessing Model Performance

Next week:

- Supervised Learning Methods: Lasso, Ridge, Support Vector Machines, and ReadMe
- Ensemble methods: combining the results of many supervised algorithms
- Cross validation: Replicate classification exercise, with data
- Avoid over training data: Balance bias and variance in model selection
- Super learning: optimal ensemble methods

Methods generalize beyond text
Supervised Learning

Today:
- How to generate valid hand coding categories
Supervised Learning

Today:
- How to generate valid hand coding categories
  - Assessing coder performance
  - Assessing disagreement among coders
  - Evidence coders perform well

Supervised Learning Methods: Naive Bayes

Next week:
- Supervised Learning Methods: Lasso, Ridge, Support Vector Machines, and ReadMe
- Ensemble methods: combining the results of many supervised algorithms
- Cross validation:
  - Replicate classification exercise, with data
  - Avoid over training data: Balance bias and variance in model selection
- Super learning: optimal ensemble methods

Methods generalize beyond text

Justin Grimmer (Stanford University)
Supervised Learning

Today:
- How to generate valid hand coding categories
  - Assessing coder performance
  - Assessing disagreement among coders
  - Evidence coders perform well
- Supervised Learning Methods: Naive Bayes

Next week:
- Supervised Learning Methods: Lasso, Ridge, Support Vector Machines, and ReadMe
- Ensemble methods: combining the results of many supervised algorithms
- Cross validation: Replicate classification exercise, with data
- Avoid over training data: Balance bias and variance in model selection
- Super learning: optimal ensemble methods

Methods generalize beyond text

Justin Grimmer (Stanford University)
Supervised Learning

Today:
- How to generate valid hand coding categories
  - Assessing coder performance
  - Assessing disagreement among coders
  - Evidence coders perform well
- Supervised Learning Methods: Naive Bayes
- Assessing Model Performance

Next week:
- Supervised Learning Methods: Lasso, Ridge, Support Vector Machines, and ReadMe
- Ensemble methods: combining the results of many supervised algorithms
- Cross validation:
  - Replicate classification exercise, with data
  - Avoid over training data: Balance bias and variance in model selection
- Super learning: optimal ensemble methods

Methods generalize beyond text

Justin Grimmer (Stanford University)
Supervised Learning

Today:
- How to generate valid hand coding categories
  - Assessing coder performance
  - Assessing disagreement among coders
  - Evidence coders perform well
- Supervised Learning Methods: Naive Bayes
- Assessing Model Performance

Next week:
- Supervised Learning Methods: Lasso, Ridge, Support Vector Machines, and ReadMe
- Ensemble methods: combining the results of many supervised algorithms
- Cross validation: Replicate classification exercise, with data
- Avoid over training data: Balance bias and variance in model selection
- Super learning: optimal ensemble methods

Methods generalize beyond text
Supervised Learning

Today:

- How to generate valid hand coding categories
  - Assessing coder performance
  - Assessing disagreement among coders
  - Evidence coders perform well
- Supervised Learning Methods: Naive Bayes
- Assessing Model Performance

Next week:

- Supervised Learning Methods: Lasso, Ridge, Support Vector Machines, and ReadMe
Supervised Learning

Today:
- How to generate valid hand coding categories
  - Assessing coder performance
  - Assessing disagreement among coders
  - Evidence coders perform well
- Supervised Learning Methods: Naive Bayes
- Assessing Model Performance

Next week:
- Supervised Learning Methods: Lasso, Ridge, Support Vector Machines, and ReadMe
- Ensemble methods: combining the results of many supervised algorithms
Supervised Learning

Today:
- How to generate valid hand coding categories
  - Assessing coder performance
  - Assessing disagreement among coders
  - Evidence coders perform well
- Supervised Learning Methods: Naive Bayes
- Assessing Model Performance

Next week:
- Supervised Learning Methods: Lasso, Ridge, Support Vector Machines, and ReadMe
- Ensemble methods: combining the results of many supervised algorithms
- Cross validation:
Supervised Learning

Today:
- How to generate valid hand coding categories
  - Assessing coder performance
  - Assessing disagreement among coders
  - Evidence coders perform well
- Supervised Learning Methods: Naive Bayes
- Assessing Model Performance

Next week:
- Supervised Learning Methods: Lasso, Ridge, Support Vector Machines, and ReadMe
- Ensemble methods: combining the results of many supervised algorithms
- Cross validation:
  - Replicate classification exercise, with data
  - Avoid over training data: Balance bias and variance in model selection
  - Super learning: optimal ensemble methods
Supervised Learning

Today:
- How to generate valid hand coding categories
  - Assessing coder performance
  - Assessing disagreement among coders
  - Evidence coders perform well
- Supervised Learning Methods: Naive Bayes
- Assessing Model Performance

Next week:
- Supervised Learning Methods: Lasso, Ridge, Support Vector Machines, and ReadMe
- Ensemble methods: combining the results of many supervised algorithms
- Cross validation:
  - Replicate classification exercise, with data
  - Avoid over training data: Balance bias and variance in model selection
  - Super learning: optimal ensemble methods

Methods generalize beyond text
Components to Supervised Learning Method

1) Set of categories
   - Credit Claiming, Position Taking, Advertising
   - Positive Tone, Negative Tone
   - Pro-war, Ambiguous, Anti-war

2) Set of hand-coded documents
   - Coding done by human coders
   - Training Set: documents we’ll use to learn how to code
   - Validation Set: documents we’ll use to learn how well we code

3) Set of unlabeled documents

4) Method to extrapolate from hand coding to unlabeled documents
Components to Supervised Learning Method

1) Set of categories

- Credit Claiming, Position Taking, Advertising
- Positive Tone, Negative Tone
- Pro-war, Ambiguous, Anti-war

2) Set of hand-coded documents

- Coding done by human coders
- Training Set: documents we'll use to learn how to code
- Validation Set: documents we'll use to learn how well we code

3) Set of unlabeled documents

4) Method to extrapolate from hand coding to unlabeled documents
Components to Supervised Learning Method

1) Set of categories
   - Credit Claiming, Position Taking, Advertising
   - Positive Tone, Negative Tone
   - Pro-war, Ambiguous, Anti-war
Components to Supervised Learning Method

1) Set of categories
   - Credit Claiming, Position Taking, Advertising
   - Positive Tone, Negative Tone
   - Pro-war, Ambiguous, Anti-war

2) Set of hand-coded documents
Components to Supervised Learning Method

1) Set of categories
   - Credit Claiming, Position Taking, Advertising
   - Positive Tone, Negative Tone
   - Pro-war, Ambiguous, Anti-war

2) Set of hand-coded documents
   - Coding done by human coders
   - Training Set: documents we’ll use to learn how to code
   - Validation Set: documents we’ll use to learn how well we code
Components to Supervised Learning Method

1) Set of **categories**
   - Credit Claiming, Position Taking, Advertising
   - Positive Tone, Negative Tone
   - Pro-war, Ambiguous, Anti-war

2) Set of **hand-coded** documents
   - Coding done by human coders
   - **Training** Set: documents we’ll use to learn how to code
   - **Validation** Set: documents we’ll use to learn how well we code

3) Set of **unlabeled** documents
Components to Supervised Learning Method

1) Set of categories
   - Credit Claiming, Position Taking, Advertising
   - Positive Tone, Negative Tone
   - Pro-war, Ambiguous, Anti-war

2) Set of hand-coded documents
   - Coding done by human coders
   - Training Set: documents we’ll use to learn how to code
   - Validation Set: documents we’ll use to learn how well we code

3) Set of unlabeled documents

4) Method to extrapolate from hand coding to unlabeled documents
How Do We Generate Coding Rules and Categories?

Challenge: coding rules/training coders to maximize coder performance

Challenge: developing a clear set of categories

1) Limits of Humans:
   - Small working memories
   - Easily distracted
   - Insufficient motivation

2) Limits of Language:
   - Fundamental ambiguity in language [careful analysis of texts]
   - Contextual nature of language

For supervised methods to work: maximize coder agreement

1) Write careful (and brief) coding rules
   - Flow charts help simplify problems

2) Train coders to remove ambiguity, misinterpretation
How Do We Generate Coding Rules and Categories?

**Challenge:** coding rules/training coders to maximize coder performance

1) **Limits of Humans:**
   - Small working memories
   - Easily distracted
   - Insufficient motivation

2) **Limits of Language:**
   - Fundamental ambiguity in language [careful analysis of texts]
   - Contextual nature of language

For supervised methods to work: maximize coder agreement

1) Write careful (and brief) coding rules
   - Flow charts help simplify problems

2) Train coders to remove ambiguity, misinterpretation
How Do We Generate Coding Rules and Categories?

**Challenge:** coding rules/training coders to maximize coder performance

**Challenge:** developing a clear set of categories
How Do We Generate Coding Rules and Categories?

**Challenge:** coding rules/training coders to maximize coder performance

**Challenge:** developing a clear set of categories

1) Limits of Humans:

- Small working memories
- Easily distracted
- Insufficient motivation

2) Limits of Language:

- Fundamental ambiguity in language
- Contextual nature of language

For supervised methods to work: maximize coder agreement

1) Write careful (and brief) coding rules
- Flow charts help simplify problems

2) Train coders to remove ambiguity, misinterpretation
How Do We Generate Coding Rules and Categories?

**Challenge:** coding rules/training coders to maximize coder performance

**Challenge:** developing a clear set of categories

1) **Limits of Humans:**
   - Small working memories
   - Easily distracted
   - Insufficient motivation

2) **Limits of Language:**
   - Fundamental ambiguity in language [careful analysis of texts]
   - Contextual nature of language

For supervised methods to work: maximize coder agreement

1) Write careful (and brief) coding rules
   - Flow charts help simplify problems

2) Train coders to remove ambiguity, misinterpretation
How Do We Generate Coding Rules and Categories?

**Challenge:** coding rules/training coders to maximize coder performance

**Challenge:** developing a clear set of categories

1) **Limits of Humans:**
   - Small working memories
   - Easily distracted
   - Insufficient motivation

2) **Limits of Language:**

Justin Grimmer (Stanford University)

Text as Data

November 6th, 2014
How Do We Generate Coding Rules and Categories?

Challenge: coding rules/training coders to maximize coder performance
Challenge: developing a clear set of categories

1) Limits of Humans:
   - Small working memories
   - Easily distracted
   - Insufficient motivation

2) Limits of Language:
   - Fundamental ambiguity in language [careful analysis of texts]
   - Contextual nature of language
How Do We Generate Coding Rules and Categories?

Challenge: coding rules/training coders to maximize coder performance

Challenge: developing a clear set of categories

1) Limits of Humans:
   - Small working memories
   - Easily distracted
   - Insufficient motivation

2) Limits of Language:
   - Fundamental ambiguity in language [careful analysis of texts]
   - Contextual nature of language

For supervised methods to work: maximize coder agreement
How Do We Generate Coding Rules and Categories?

Challenge: coding rules/training coders to maximize coder performance
Challenge: developing a clear set of categories

1) Limits of Humans:
   - Small working memories
   - Easily distracted
   - Insufficient motivation

2) Limits of Language:
   - Fundamental ambiguity in language [careful analysis of texts]
   - Contextual nature of language

For supervised methods to work: maximize coder agreement

1) Write careful (and brief) coding rules
How Do We Generate Coding Rules and Categories?

**Challenge:** coding rules/training coders to maximize coder performance

**Challenge:** developing a clear set of categories

1) **Limits of Humans:**
   - Small working memories
   - Easily distracted
   - Insufficient motivation

2) **Limits of Language:**
   - Fundamental ambiguity in language [careful analysis of texts]
   - Contextual nature of language

For supervised methods to work: maximize coder agreement

1) Write careful (and brief) coding rules
   - Flow charts help simplify problems
How Do We Generate Coding Rules and Categories?

**Challenge:** coding rules/training coders to maximize coder performance

**Challenge:** developing a clear set of categories

1) **Limits of Humans:**
   - Small working memories
   - Easily distracted
   - Insufficient motivation

2) **Limits of Language:**
   - Fundamental ambiguity in language [careful analysis of texts]
   - Contextual nature of language

For supervised methods to work: maximize coder agreement

1) **Write careful (and brief) coding rules**
   - Flow charts help simplify problems

2) **Train coders to remove ambiguity, misinterpretation**
How Do We Generate Coding Rules?

Iterative process for generating coding rules:

1) Write a set of coding rules
2) Have coders code documents (about 200)
3) Assess coder agreement
4) Identify sources of disagreement, repeat
Iterative process for generating coding rules:

1) Write a set of coding rules
How Do We Generate Coding Rules?

Iterative process for generating coding rules:

1) Write a set of coding rules
2) Have coders code documents (about 200)
How Do We Generate Coding Rules?

Iterative process for generating coding rules:
1) Write a set of coding rules
2) Have coders code documents (about 200)
3) Assess coder agreement
Iterative process for generating coding rules:

1) Write a set of coding rules
2) Have coders code documents (about 200)
3) Assess coder agreement
4) Identify sources of disagreement, repeat
How Do We Identify Coding Disagreement?

Many measures of inter-coder agreement
Essentially attempt to summarize a confusion matrix

<table>
<thead>
<tr>
<th></th>
<th>Cat 1</th>
<th>Cat 2</th>
<th>Cat 3</th>
<th>Cat 4</th>
<th>Sum, Coder 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cat 1</td>
<td>30</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>31</td>
</tr>
<tr>
<td>Cat 2</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>Cat 3</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Cat 4</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>7</td>
<td>11</td>
</tr>
<tr>
<td>Sum, Coder 2</td>
<td>34</td>
<td>2</td>
<td>2</td>
<td>7</td>
<td>Total: 45</td>
</tr>
</tbody>
</table>

- **Diagonal**: coders agree on document
- **Off-diagonal**: coders disagree (confused) on document

Generalize across \((k)\) coders:

- \(\frac{k(k-1)}{2}\) pairwise comparisons
- \(k\) comparisons: Coder A against All other coders
How Do We Identify Coding Disagreements?

During coding development phase/coder assessment phase, full confusion matrices help to identify
- Ambiguity
- Coder slacking

Example: 3 Coders, 8 categories.
How Do We Identify Coding Disagreements?

During coding development phase/coder assessment phase, full confusion matrices help to identify

- Ambiguity
- Coder slacking

Example: 3 Coders, 8 categories.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coder B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>15</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>13</td>
<td>7</td>
<td>0</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>11</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>32</td>
<td>0</td>
<td>1</td>
<td>36</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>13</td>
<td>26</td>
<td>36</td>
<td>36</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>7</td>
<td>0</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>Total</td>
<td>30</td>
<td>3</td>
<td>4</td>
<td>10</td>
<td>15</td>
<td>63</td>
<td>27</td>
<td>8</td>
<td>47</td>
</tr>
</tbody>
</table>
How Do We Identify Coding Disagreements?

During coding development phase/coder assessment phase, full confusion matrices help to identify

- Ambiguity
- Coder slacking

Example: 3 Coders, 8 categories.
How Do We Identify Coding Disagreements?

During coding development phase/coder assessment phase, full confusion matrices help to identify

- Ambiguity
- Coder slacking

Example: 3 Coders, 8 categories.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coder B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>18</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>7</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>18</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>13</td>
<td>1</td>
<td>7</td>
<td>4</td>
<td>1</td>
<td>26</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>1</td>
<td>7</td>
<td>4</td>
<td>1</td>
<td>26</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>1</td>
<td>15</td>
</tr>
<tr>
<td>Total</td>
<td>35</td>
<td>1</td>
<td>10</td>
<td>13</td>
<td>19</td>
<td>42</td>
<td>64</td>
<td>17</td>
<td></td>
</tr>
</tbody>
</table>
Example Coding Document

8 part coding scheme

- **Across Party Taunting**: explicit public and negative attacks on the other party or its members
- **Within Party Taunting**: explicit public and negative attacks on the same party or its members [for 1960’s politics]
- **Other taunting**: explicit public and negative attacks not directed at a party
- **Bipartisan support**: praise for the other party
- **Honorary Statements**: qualitatively different kind of speech
- **Policy speech**: a speech without taunting or credit claiming
- **Procedural**
- **No Content**: (occasionally occurs in CR)
How Do We Summarize Confusion Matrix?

Lots of statistics to summarize confusion matrix:

- **Most common**: intercoder agreement

$$\text{Inter Coder}(A, B) = \frac{\text{No. (Coder A & Coder B agree)}}{\text{No. Documents}}$$
Liberal measure of agreement:

- Some agreement by chance
- Consider coding scheme with two categories
  \{ Class 1, Class 2 \}
- Coder A and Coder B flip a (biased coin).
  \( \Pr(\text{Class 1}) = 0.75, \Pr(\text{Class 2}) = 0.25 \)
- Inter Coder reliability: 0.625

What to do?
Suggestion: Subtract off amount expected by chance:

\[
\text{Inter Coder}(A, B) = \frac{\text{No. (Coder A & Coder B agree)} - \text{No. Expected by Chance}}{\text{No. Documents}}
\]

Question: what is amount expected by chance?
- \#Categories?
- Avg Proportion in categories across coders? (Krippendorf's Alpha)

Best Practice: present confusion matrices.

Justin Grimmer (Stanford University) Text as Data November 6th, 2014 13 / 34
Liberal measure of agreement:
- Some agreement by chance

- Consider coding scheme with two categories {Class 1, Class 2}.
- Coder A and Coder B flip a (biased) coin. (Pr(Class 1) = 0.75, Pr(Class 2) = 0.25)
- Inter Coder reliability: 0.625

What to do?
Suggestion: Subtract off amount expected by chance:

\[ \text{Inter Coder}(A, B) \text{ norm} = \frac{\text{No. (Coder A & Coder B agree)} - \text{No. Expected by Chance}}{\text{No. Documents}} \]

Question: what is amount expected by chance?
- 1 #Categories?
- Avg Proportion in categories across coders? (Krippendorf's Alpha)

Best Practice: present confusion matrices.

Justin Grimmer (Stanford University)
Liberal measure of agreement:
  - Some agreement by chance
  - Consider coding scheme with two categories
    \{ Class 1, Class 2 \}.
Liberal measure of agreement:
- Some agreement by chance
- Consider coding scheme with two categories
  \{ Class 1, Class 2 \}.
- Coder A and Coder B flip a (biased coin).
  ( Pr(Class 1) = 0.75, Pr(Class 2) = 0.25 )
Liberal measure of agreement:

- Some agreement by *chance*
- Consider coding scheme with two categories
  \{ Class 1, Class 2 \}.
- Coder A and Coder B flip a (biased coin).
  \( \Pr(\text{Class 1}) = 0.75, \Pr(\text{Class 2}) = 0.25 \)
- Inter Coder reliability: 0.625

What to do?

Suggestion: Subtract off amount expected by chance:

\[
\text{Inter Coder}(\text{A}, \text{B}) \text{norm} = \frac{\text{No. (Coder A & Coder B agree)}}{\text{No. Documents}} - \text{No. Expected by Chance}
\]

Question: what is amount expected by chance?

- \#Categories?
- Avg Proportion in categories across coders? (Krippendorf’s Alpha)

Best Practice: present confusion matrices.
Liberal measure of agreement:

- Some agreement by chance
- Consider coding scheme with two categories \{ Class 1, Class 2 \}.
- Coder A and Coder B flip a (biased coin).
  \( \Pr(\text{Class 1}) = 0.75, \Pr(\text{Class 2}) = 0.25 \)
- Inter Coder reliability: 0.625

What to do?

Suggestion: Subtract off amount expected by chance:
\[
\text{Inter Coder}(A, B)_{\text{norm}} = \frac{\text{No. (Coder A & Coder B agree)}}{\text{No. Documents}} - \frac{\text{No. Expected by Chance}}{\text{No. Documents}}
\]

Question: what is amount expected by chance?
- \(1 \# \text{Categories}\)?
- Avg Proportion in categories across coders? (Krippendorf's Alpha)

Best Practice: present confusion matrices.
Liberal measure of agreement:

- Some agreement by chance
- Consider coding scheme with two categories
  \{ Class 1, Class 2 \}.
- Coder A and Coder B flip a (biased coin).
  \( \Pr(\text{Class 1}) = 0.75, \Pr(\text{Class 2}) = 0.25 \)
- Inter Coder reliability: 0.625

What to do?
Suggestion: Subtract off amount expected by chance:
Liberal measure of agreement:

- Some agreement by chance
- Consider coding scheme with two categories \{ Class 1, Class 2 \}.
- Coder A and Coder B flip a (biased coin).
  \( \text{Pr(Class 1) = 0.75, Pr(Class 2) = 0.25} \)
- Inter Coder reliability: 0.625

What to do?

Suggestion: Subtract off amount expected by chance:

\[
\text{Inter Coder}(A, B)_{\text{norm}} = \frac{\text{No. (Coder A & Coder B agree)} - \text{No. Expected by Chance}}{\text{No. Documents}}
\]

Question: what is amount expected by chance?

- \#Categories?
- Avg Proportion in categories across coders? (Krippendorf's Alpha)

Best Practice: present confusion matrices.
Liberal measure of agreement:

- Some agreement by chance
- Consider coding scheme with two categories \{ Class 1, Class 2 \}.
- Coder A and Coder B flip a (biased coin).
  \( \Pr(\text{Class 1}) = 0.75, \Pr(\text{Class 2}) = 0.25 \)
- Inter Coder reliability: 0.625

What to do?

Suggestion: Subtract off amount expected by chance:

\[
\text{Inter Coder}(A, B)_{\text{norm}} = \frac{\text{No. (Coder A & Coder B agree)} - \text{No. Expected by Chance}}{\text{No. Documents}}
\]

Question: what is amount expected by chance?
Liberal measure of agreement:
- Some agreement by chance
- Consider coding scheme with two categories 
  \{ Class 1, Class 2 \}.
- Coder A and Coder B flip a (biased coin).
  \( \Pr(\text{Class 1}) = 0.75, \Pr(\text{Class 2}) = 0.25 \) 
- Inter Coder reliability: 0.625

What to do?
Suggestion: Subtract off amount expected by chance:

\[
\text{Inter Coder}(A, B)_{\text{norm}} = \frac{\text{No. (Coder A & Coder B agree) } - \text{No. Expected by Chance}}{\text{No. Documents}}
\]

Question: what is amount expected by chance?
- \( \frac{1}{\#\text{Categories}} \)?
- Avg Proportion in categories across coders? (Krippendorf’s Alpha)
Liberal measure of agreement:
- Some agreement by chance
- Consider coding scheme with two categories { Class 1, Class 2 }.
- Coder A and Coder B flip a (biased coin).
  ( Pr(Class 1) = 0.75, Pr(Class 2) = 0.25 )
- Inter Coder reliability: 0.625

What to do?
Suggestion: Subtract off amount expected by chance:

\[
\text{Inter Coder}(A, B)_{\text{norm}} = \frac{\text{No. (Coder A & Coder B agree)} - \text{No. Expected by Chance}}{\text{No. Documents}}
\]

Question: what is amount expected by chance?
- \( \frac{1}{\text{#Categories}} \) ?
- Avg Proportion in categories across coders? (Krippendorf’s Alpha)

Best Practice: present confusion matrices.
Krippendorf’s Alpha

Define coder reliability as:

\[ \alpha = 1 - \frac{\text{No. Pairwise Disagreements Observed}}{\text{No. Pairwise Disagreements Expected By Chance}} \]

No. Pairwise Disagreements Observed = observe from data

No Expected pairwise disagreements: coding by chance, with rate labels used available from data

Thinking through expected differences:

- Pretend I know something I'm trying to estimate
- How is that we know coders estimate levels well?
- Have to present correlation statistic: vary assumptions about "expectations" (from uniform, to data driven)

Calculate in \texttt{R} with \texttt{concord} package and function \texttt{kripp.alpha}

Justin Grimmer (Stanford University)
Krippendorf’s Alpha

Define coder reliability as:

$$\alpha = 1 - \frac{\text{No. Pairwise Disagreements Observed}}{\text{No Pairwise Disagreements Expected By Chance}}$$
Krippendorf’s Alpha

Define coder reliability as:

\[ \alpha = 1 - \frac{\text{No. Pairwise Disagreements Observed}}{\text{No Pairwise Disagreements Expected By Chance}} \]

No. Pairwise Disagreements Observed = observe from data
Krippendorf’s Alpha

Define coder reliability as:

\[ \alpha = 1 - \frac{\text{No. Pairwise Disagreements Observed}}{\text{No Pairwise Disagreements Expected By Chance}} \]

No. Pairwise Disagreements Observed = observe from data
No Expected pairwise disagreements: coding by chance, with rate labels used available from data

Thinking through expected differences:
- Pretend I know something I’m trying to estimate
- How is that we know coders estimate levels well?
- Have to present correlation statistic: vary assumptions about “expectations” (from uniform, to data driven)

Calculate in R with concord package and function kripp.alpha

Justin Grimmer (Stanford University)
Krippendorf’s Alpha

Define coder reliability as:

$$\alpha = 1 - \frac{\text{No. Pairwise Disagreements Observed}}{\text{No Pairwise Disagreements Expected By Chance}}$$

- No. Pairwise Disagreements Observed = observe from data
- No Expected pairwise disagreements: coding by chance, with rate labels used available from data

Thinking through expected differences:
Krippendorf’s Alpha

Define coder reliability as:

\[ \alpha = 1 - \frac{\text{No. Pairwise Disagreements Observed}}{\text{No Pairwise Disagreements Expected By Chance}} \]

- No. Pairwise Disagreements Observed = observe from data
- No Expected pairwise disagreements: coding by chance, with rate labels used available from data

Thinking through expected differences:

- Pretend I know something I’m trying to estimate
- How is that we know coders estimate levels well?
- Have to present correlation statistic: vary assumptions about “expectations” (from uniform, to data driven)
Krippendorf’s Alpha

Define coder reliability as:

\[ \alpha = 1 - \frac{\text{No. Pairwise Disagreements Observed}}{\text{No Pairwise Disagreements Expected By Chance}} \]

No. Pairwise Disagreements Observed = observe from data
No Expected pairwise disagreements: coding by chance, with rate labels used available from data

Thinking through expected differences:
- Pretend I know something I’m trying to estimate
- How is that we know coders estimate levels well?
- Have to present correlation statistic: vary assumptions about “expectations” (from uniform, to data driven)

Calculate in R with concord package and function kripp.alpha
How Many To Code By Hand/How Many to Code By Machine

Next week: we’ll discuss how to answer this question systematically for your data set.

Rules of thumb:

- Hopkins and King (2010): 500 documents likely sufficient
- Hopkins and King (2010): 100 documents may be enough
- **BUT**: depends on quantity of interest
- May **REQUIRE** many more documents
Percent data coded, Error (From Dan Jurafsky)

Training size

Figure 2: Test error vs training size on the newsgroups alt.atheism and talk.religion.misc
Three categories of documents

Hand labeled
- Training set (what we’ll use to estimate model)
- Validation set (what we’ll use to assess model)

Unlabeled
- Test set (what we’ll use the model to categorize)

Label more documents than necessary to train model
Methods to Perform Supervised Classification

- Use the hand labels to train a statistical model.
- Naive Bayes
  - Shockingly simple application of Bayes’ rule
  - Shockingly useful \(\rightarrow\) often default classifier
Naive Bayes and General Problem Setup

Suppose we have document \( i \), \( (i = 1, \ldots, N) \) with \( J \) features.
Naive Bayes and General Problem Setup

Suppose we have document \( i, (i = 1, \ldots, N) \) with \( J \) features
\( \mathbf{x}_i = (x_{1i}, x_{2i}, \ldots, x_{Ji}) \)
Naive Bayes and General Problem Setup

Suppose we have document \( i, (i = 1, \ldots, N) \) with \( J \) features \( x_i = (x_{1i}, x_{2i}, \ldots, x_{Ji}) \)

Set of \( K \) categories. Category \( k (k = 1, \ldots, K) \)
\{ \( C_1, C_2, \ldots, C_K \) \}
Naive Bayes and General Problem Setup

Suppose we have document $i$, $(i = 1, \ldots, N)$ with $J$ features $x_i = (x_{1i}, x_{2i}, \ldots, x_{ji})$

Set of $K$ categories. Category $k$ $(k = 1, \ldots, K)$

$\{C_1, C_2, \ldots, C_K\}$

Subset of labeled documents $Y = (Y_1, Y_2, \ldots, Y_{N_{\text{train}}})$ where $Y_i \in \{C_1, C_2, \ldots, C_K\}$. 
Naive Bayes and General Problem Setup

Suppose we have document $i$, ($i = 1, \ldots, N$) with $J$ features

$$\mathbf{x}_i = (x_{1i}, x_{2i}, \ldots, x_{JI})$$

Set of $K$ categories. Category $k$ ($k = 1, \ldots, K$)

$$\{C_1, C_2, \ldots, C_K\}$$

Subset of labeled documents $\mathbf{Y} = (Y_1, Y_2, \ldots, Y_{N_{\text{train}}})$ where

$$Y_i \in \{C_1, C_2, \ldots, C_K\}.$$ 

Goal: classify every document into one category.
Naive Bayes and General Problem Setup

Suppose we have document $i$, ($i = 1, \ldots, N$) with $J$ features

$x_i = (x_{1i}, x_{2i}, \ldots, x_{Ji})$

Set of $K$ categories. Category $k$ ($k = 1, \ldots, K$)

$\{ C_1, C_2, \ldots, C_K \}$

Subset of labeled documents $Y = (Y_1, Y_2, \ldots, Y_{N_{train}})$ where

$Y_i \in \{ C_1, C_2, \ldots, C_K \}$.

**Goal:** classify every document into one category.

Learn a function that maps from space of (possible) documents to categories
Naive Bayes and General Problem Setup

Suppose we have document $i$, $(i = 1, \ldots, N)$ with $J$ features $\mathbf{x}_i = (x_{1i}, x_{2i}, \ldots, x_{Ji})$

Set of $K$ categories. Category $k$ $(k = 1, \ldots, K)$

\{ $C_1, C_2, \ldots, C_K$ \}

Subset of labeled documents $\mathbf{Y} = (Y_1, Y_2, \ldots, Y_{N_{\text{train}}})$ where $Y_i \in \{ C_1, C_2, \ldots, C_K \}$.

**Goal:** classify every document into one category.

Learn a function that maps from space of (possible) documents to categories

To do this: use hand coded observations to estimate (train) regression model
Naive Bayes and General Problem Setup

Suppose we have document \( i \), \((i = 1, \ldots, N)\) with \( J \) features
\[ x_i = (x_{1i}, x_{2i}, \ldots, x_{Ji}) \]
Set of \( K \) categories. Category \( k \) \((k = 1, \ldots, K)\)
\{\( C_1, C_2, \ldots, C_K \}\)
Subset of labeled documents \( Y = (Y_1, Y_2, \ldots, Y_{N_{\text{train}}}) \) where
\( Y_i \in \{C_1, C_2, \ldots, C_K\}\).

Goal: classify every document into one category.
Learn a function that maps from space of (possible) documents to categories
To do this: use hand coded observations to estimate (train) regression model
Apply model to test data, classify those observations
Naive Bayes and General Problem Setup (Jurafsky Inspired Slide)

Goal: For each document $x_i$, we want to infer most likely category $C_{\text{Max}}$.

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

We're going to use Bayes' rule to estimate $p(C_k | x_i)$.

$p(C_k | x_i) = \frac{p(C_k, x_i)}{p(x_i)} = \frac{p(C_k) p(x_i | C_k)}{p(x_i)}$  \hspace{1cm} (1)
Goal: For each document $x_i$, we want to infer most likely category

$$C_{\text{Max}} = \arg \max_k p(C_k | x_i)$$
Naive Bayes and General Problem Setup (Jurafsky Inspired Slide)

Goal: For each document $x_i$, we want to infer most likely category

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

We’re going to use Bayes’ rule to estimate $p(C_k|x_i)$. (1)
Naive Bayes and General Problem Setup (Jurafsky Inspired Slide)

Goal: For each document $x_i$, we want to infer most likely category

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

We’re going to use Bayes’ rule to estimate $p(C_k|x_i)$.

$$p(C_k|x_i) = \frac{p(C_k, x_i)}{p(x_i)}$$

(1)
Naive Bayes and General Problem Setup (Jurafsky Inspired Slide)

Goal: For each document $x_i$, we want to infer most likely category

$$C_{\text{Max}} = \text{arg max}_k p(C_k | x_i)$$

We’re going to use Bayes’ rule to estimate $p(C_k | x_i)$.

$$p(C_k | x_i) = \frac{p(C_k, x_i)}{p(x_i)} = \frac{p(C_k) p(x_i | C_k)}{p(x_i)}$$ (1)
Naive Bayes and General Problem Setup (Jurafsky Inspired Slide)

Goal: For each document $x_i$, we want to infer most likely category

$$C_{\text{Max}} = \text{arg max}_k p(C_k|x_i)$$

We’re going to use Bayes’ rule to estimate $p(C_k|x_i)$.

$$p(C_k|x_i) = \frac{p(C_k, x_i)}{p(x_i)}$$

Proportion in $C_k$

$$= \frac{\underbrace{p(C_k)}_{\text{Language model}} \underbrace{p(x_i|C_k)}_{\text{Language model}}}{p(x_i)}$$
Naive Bayes and Optimization (Jurafsky Inspired Slide)

Max = \arg \max_k \ p(C_k | x_i)

\begin{align*}
\text{Two probabilities to estimate:} \\
p(C_k) &= \frac{\text{No. Documents in } k}{\text{No. Documents (training set)}} \\
p(x_i | C_k) &\text{ complicated without assumptions} \\
&\text{Imagine each } x_{ij} \text{ just binary indicator. Then } 2^J \text{ possible documents} \\
&\text{Simplify: assume each feature is independent} \\
p(x_i | C_k) &= J \prod_{j=1}^J p(x_{ij} | C_k)
\end{align*}
Naive Bayes and Optimization (Jurafsky Inspired Slide)

\[ C_{\text{Max}} = \arg \max_k p(C_k | x_i) \]
Naive Bayes and Optimization (Jurafsky Inspired Slide)

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

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

Two probabilities to estimate:

$p(C_k) = \frac{\text{No. Documents in } k}{\text{No. Documents (training set)}}$

$p(x_i|C_k)$ complicated without assumptions
- Imagine each $x_{ij}$ just binary indicator. Then $2^J$ possible $x_i$ documents
- Simplify: assume each feature is independent

$p(x_i|C_k) = \prod_{j=1}^{J} p(x_{ij}|C_k)$
Naive Bayes and Optimization (Jurafsky Inspired Slide)

\[ C_{\text{Max}} = \arg \max_k p(C_k | x_i) \]

\[ C_{\text{Max}} = \arg \max_k \frac{p(C_k)p(x_i | C_k)}{p(x_i)} \]

\[ C_{\text{Max}} = \arg \max_k p(C_k)p(x_i | C_k) \]

Two probabilities to estimate:

- \( p(C_k) = \frac{\text{No. Documents in } k}{\text{No. Documents (training set)}} \)
- \( p(x_i | C_k) \) complicated without assumptions
  - Imagine each \( x_{ij} \) just binary indicator. Then \( 2^J \) possible \( x \) documents
  - Simplify: assume each feature is independent
    \[ p(x_i | C_k) = \prod_{j=1}^{J} p(x_{ij} | C_k) \]
Naive Bayes and Optimization (Jurafsky Inspired Slide)

\[
C_{\text{Max}} = \arg \max_k p(C_k | x_i)
\]
\[
C_{\text{Max}} = \arg \max_k \frac{p(C_k)p(x_i | C_k)}{p(x_i)}
\]
\[
C_{\text{Max}} = \arg \max_k p(C_k)p(x_i | C_k)
\]

Two probabilities to estimate:
Naive Bayes and Optimization (Jurafsky Inspired Slide)

\[ C_{\text{Max}} = \arg \max_k p(C_k | x_i) \]

\[ C_{\text{Max}} = \arg \max_k \frac{p(C_k)p(x_i | C_k)}{p(x_i)} \]

\[ C_{\text{Max}} = \arg \max_k p(C_k)p(x_i | C_k) \]

Two probabilities to estimate:

\[ p(C_k) = \frac{\text{No. Documents in } k}{\text{No. Documents}} \text{ (training set)} \]
Naive Bayes and Optimization (Jurafsky Inspired Slide)

\[ C_{\text{Max}} = \arg \max_k p(C_k | x_i) \]
\[ C_{\text{Max}} = \arg \max_k \frac{p(C_k)p(x_i | C_k)}{p(x_i)} \]
\[ C_{\text{Max}} = \arg \max_k p(C_k)p(x_i | C_k) \]

Two probabilities to estimate:

\[ p(C_k) = \frac{\text{No. Documents in } k}{\text{No. Documents}} \text{ (training set)} \]
\[ p(x_i | C_k) \text{ complicated without assumptions} \]
Naive Bayes and Optimization (Jurafsky Inspired Slide)

\[ C_{\text{Max}} = \arg \max_k p(C_k|x_i) \]
\[ C_{\text{Max}} = \arg \max_k \frac{p(C_k)p(x_i|C_k)}{p(x_i)} \]
\[ C_{\text{Max}} = \arg \max_k p(C_k)p(x_i|C_k) \]

Two probabilities to estimate:

\( p(C_k) = \frac{\text{No. Documents in } k}{\text{No. Documents}} \) (training set)

\( p(x_i|C_k) \) complicated without assumptions

- Imagine each \( x_{ij} \) just binary indicator. Then \( 2^J \) possible \( x_i \) documents
Naive Bayes and Optimization (Jurafsky Inspired Slide)

\[
C_{\text{Max}} = \arg \max_k p(C_k \mid x_i)
\]

\[
C_{\text{Max}} = \arg \max_k \frac{p(C_k)p(x_i \mid C_k)}{p(x_i)}
\]

\[
C_{\text{Max}} = \arg \max_k p(C_k)p(x_i \mid C_k)
\]

Two probabilities to estimate:

\[
p(C_k) = \frac{\text{No. Documents in } k}{\text{No. Documents in training set}}
\]

\[
p(x_i \mid C_k) \text{ complicated without assumptions}
\]

- Imagine each \( x_{ij} \) just binary indicator. Then \( 2^J \) possible \( x_i \) documents
- Simplify: assume each feature is independent
$$C_{\text{Max}} = \arg \max_k p(C_k | x_i)$$

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

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

Two probabilities to estimate:

$$p(C_k) = \frac{\text{No. Documents in } k}{\text{No. Documents}} \text{ (training set)}$$

$$p(x_i | C_k) \text{ complicated without assumptions}$$

- Imagine each $x_{ij}$ just binary indicator. Then $2^J$ possible $x_i$ documents
- Simplify: assume each feature is independent

$$p(x_i | C_k) = \prod_{j=1}^{J} p(x_{ij} | C_k)$$
Naive Bayes and Optimization (Jurafsky Inspired Slide)

Two components to estimation:

- $p(C_k) = \frac{\text{No. Documents in } k}{\text{No. Documents}}$ (training set)

- $p(x_i | C_k) = \prod_{j=1}^J p(x_{ij} | C_k)$
Two components to estimation:

- \( p(C_k) = \frac{\text{No. Documents in } k}{\text{No. Documents}} \)  (training set)

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

Maximum likelihood estimation (training set):

\( p(x_{im} = z | C_k) = \frac{\text{No}(\text{Docs } x_{ij} = z \text{ and } C = C_k)}{\text{No}(C = C_k)} \)

Problem: What if \( \text{No}(\text{Docs } x_{ij} = z \text{ and } C = C_k) = 0 \) ?

\( \prod_{j=1}^{J} p(x_{ij}|C_k) = 0 \)
Two components to estimation:

- \( p(C_k) = \frac{\text{No. Documents in } k}{\text{No. Documents}} \) (training set)
- \( p(x_i | C_k) = \prod_{j=1}^{J} p(x_{ij} | C_k) \)

Maximum likelihood estimation (training set):

\[
p(x_{im} = z | C_k) = \frac{\text{No( Docs}_{ij} = z \text{ and } C = C_k)}{\text{No}(C = C_k)}
\]
Naive Bayes and Optimization (Jurafsky Inspired Slide)

Two components to estimation:
- \( p(C_k) = \frac{\text{No. Documents in } k}{\text{No. Documents}} \) (training set)
- \( p(x_i|C_k) = \prod_{j=1}^{J} p(x_{ij}|C_k) \)

Maximum likelihood estimation (training set):

\[
p(x_{im} = z|C_k) = \frac{\text{No( Docs}_{ij} = z \text{ and } C = C_k )}{\text{No}(C = C_k)}
\]

Problem: What if \( \text{No( Docs}_{ij} = z \text{ and } C = C_k ) = 0 \)?
Two components to estimation:

- \( p(C_k) = \frac{\text{No. Documents in } k}{\text{No. Documents}} \) (training set)

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

Maximum likelihood estimation (training set):

\[
p(x_{im} = z | C_k) = \frac{\text{No( Docs}_{ij} = z \text{ and } C = C_k )}{\text{No( C= } C_k \text{ )}}
\]

Problem: What if \( \text{No( Docs}_{ij} = z \text{ and } C = C_k ) = 0 \) ?

\[
\prod_{j=1}^{J} p(x_{ij} | C_k) = 0
\]
Naive Bayes and General Problem Setup (Jurafsky Inspired Slide)

Solution: smoothing (Bayesian estimation)

\[ p(x_{ij} = z | C_k) = \frac{\text{No}(\text{Docs}_{ij} = z \text{ and } C = C_k)}{\text{No}(C = C_k) + k} \]

Algorithm steps:
1) Learn \( \hat{p}(C) \) and \( \hat{p}(x_i | C_k) \) on training data
2) Use this to identify most likely \( C_k \) for each document \( i \) in test set

\[ C_i = \arg \max_k \hat{p}(C_k) \hat{p}(x_i | C_k) \]

Simple intuition about Naive Bayes:
- Learn what documents in class \( j \) look like
- Find class \( k \) that document \( i \) is most similar to
Naive Bayes and General Problem Setup (Jurafsky Inspired Slide)

Solution: smoothing (Bayesian estimation)
Naive Bayes and General Problem Setup (Jurafsky Inspired Slide)

Solution: smoothing (Bayesian estimation)

\[
p(x_{ij} = z | C_k) = \frac{\text{No} (\text{Docs}_{ij} = z \text{ and } C = C_k)}{\text{No}(C = C_k) + k} + 1
\]
Naive Bayes and General Problem Setup (Jurafsky Inspired Slide)

Solution: smoothing (Bayesian estimation)

\[ p(x_{ij} = z|C_k) = \frac{\text{No( Docs}_{ij} = z \text{ and } C = C_k ) + 1}{\text{No}(C = C_k) + k} \]

Algorithm steps:
1) Learn \( \hat{p}(C_k) \) and \( \hat{p}(x_i|C_k) \) on training data
2) Use this to identify most likely \( C_k \) for each document \( i \) in test set

Simple intuition about Naive Bayes:
- Learn what documents in class \( j \) look like
- Find class \( k \) that document \( i \) is most similar to
Naive Bayes and General Problem Setup (Jurafsky Inspired Slide)

Solution: smoothing (Bayesian estimation)

\[
p(x_{ij} = z | C_k) = \frac{\text{No( Docs}_{ij} = z \text{ and } C = C_k ) + 1}{\text{No}(C = C_k) + k}
\]

Algorithm steps:

1) Learn \( \hat{p}(C) \) and \( \hat{p}(x_i | C_k) \) on training data
Naive Bayes and General Problem Setup (Jurafsky Inspired Slide)

Solution: smoothing (Bayesian estimation)

\[ p(x_{ij} = z | C_k) = \frac{\text{No}(\text{Docs}_{ij} = z \text{ and } C = C_k) + 1}{\text{No}(C = C_k) + k} \]

Algorithm steps:
1) Learn \( \hat{p}(C) \) and \( \hat{p}(x_i | C_k) \) on training data
2) Use this to identify most likely \( C_k \) for each document \( i \) in test set
Naive Bayes and General Problem Setup (Jurafsky Inspired Slide)

Solution: smoothing (Bayesian estimation)

\[
p(x_{ij} = z | C_k) = \frac{\text{No}(\text{Docs}_{ij} = z \text{ and } C = C_k) + 1}{\text{No}(C = C_k) + k}
\]

Algorithm steps:
1) Learn \( \hat{p}(C) \) and \( \hat{p}(x_i|C_k) \) on training data
2) Use this to identify most likely \( C_k \) for each document \( i \) in test set

\[
C_i = \arg \max_k \hat{p}(C_k) \hat{p}(x_i|C_k)
\]
Naive Bayes and General Problem Setup (Jurafsky Inspired Slide)

Solution: smoothing (Bayesian estimation)

\[
p(x_{ij} = z | C_k) = \frac{\text{No( Docs}_{ij} = z \text{ and } C = C_k ) + 1}{\text{No}(C= C_k) + k}
\]

Algorithm steps:
1) Learn \( \hat{p}(C) \) and \( \hat{p}(x_i | C_k) \) on training data
2) Use this to identify most likely \( C_k \) for each document \( i \) in test set

\[
C_i = \text{arg max}_k \hat{p}(C_k) \hat{p}(x_i | C_k)
\]

Simple intuition about Naive Bayes:
- Learn what documents in class \( j \) look like
- Find class \( k \) that document \( i \) is most similar to
Naive Bayes and General Problem Setup (Jurafsky Inspired Slide)

Solution: smoothing (Bayesian estimation)

\[ p(x_{ij} = z | C_k) = \frac{\text{No( Docs}_{ij} = z \text{ and } C = C_k ) + 1}{\text{No}(C = C_k) + k} \]

Algorithm steps:

1) Learn \( \hat{p}(C) \) and \( \hat{p}(x_i | C_k) \) on training data
2) Use this to identify most likely \( C_k \) for each document \( i \) in test set

\[ C_i = \arg \max_k \hat{p}(C_k) \hat{p}(x_i | C_k) \]

Simple intuition about Naive Bayes:

- Learn what documents in class \( j \) look like
Naive Bayes and General Problem Setup (Jurafsky Inspired Slide)

Solution: smoothing (Bayesian estimation)

\[
p(x_{ij} = z | C_k) = \frac{\text{No}(\text{Docs}_{ij} = z \text{ and } C = C_k)}{\text{No}(C = C_k) + k} + 1
\]

Algorithm steps:
1) Learn \( \hat{p}(C) \) and \( \hat{p}(x_i | C_k) \) on training data
2) Use this to identify most likely \( C_k \) for each document \( i \) in test set

\[
C_i = \arg \max_k \hat{p}(C_k) \hat{p}(x_i | C_k)
\]

Simple intuition about Naive Bayes:
- Learn what documents in class \( j \) look like
- Find class \( k \) that document \( i \) is most similar to
Naive Bayes and Unigram Language Models

Assume the following data generating process (should look familiar):

\[ \pi \sim \text{Dirichlet}(\alpha) \]
\[ \theta \sim \text{Dirichlet}(\lambda) \]
\[ \tau_i \sim \text{Multinomial}(1, \pi) \]
\[ x_i | \tau_{ik} = 1, \theta \sim \text{Multinomial}(n_i, \theta_k) \]
Naive Bayes and Unigram Language Models

Assume the following data generating process (should look familiar)

\[
\begin{align*}
\pi & \sim \text{Dirichlet}(\alpha) \\
\theta & \sim \text{Dirichlet}(\lambda) \\
\tau_i & \sim \text{Multinomial}(1, \pi) \\
x_i | \tau_{ik} = 1, \theta & \sim \text{Multinomial}(n_i, \theta_k)
\end{align*}
\]

If we randomly sample documents \( N_{\text{train}} \) and label them \((Y)\), then we can estimate
Naive Bayes and Unigram Language Models

Assume the following data generating process (should look familiar)

\[ \pi \sim \text{Dirichlet}(\alpha) \]
\[ \theta \sim \text{Dirichlet}(\lambda) \]
\[ \tau_i \sim \text{Multinomial}(1, \pi) \]
\[ x_i | \tau_{ik} = 1, \theta \sim \text{Multinomial}(n_i, \theta_k) \]

If we randomly sample documents \( N_{\text{train}} \) and label them \( (Y) \), then we can estimate

\[ \hat{\pi}_k = \frac{\sum_{i=1}^{N} I(Y_i = k) + \alpha_k}{N_{\text{train}}} \]
Naive Bayes and Unigram Language Models

Assume the following data generating process (should look familiar)

\[ \pi \sim \text{Dirichlet}(\alpha) \]
\[ \theta \sim \text{Dirichlet}(\lambda) \]
\[ \tau_i \sim \text{Multinomial}(1, \pi) \]
\[ x_i|\tau_{ik} = 1, \theta \sim \text{Multinomial}(n_i, \theta_k) \]

If we randomly sample documents \( N_{\text{train}} \) and label them \((Y)\), then we can estimate

\[
\hat{\pi}_k = \frac{\sum_{i=1}^{N} I(Y_i = k) + \alpha_k}{N_{\text{train}}}
\]
\[
\hat{\theta}_{jk} = \frac{\sum_{i=1}^{N} I(Y_i = k)x_{ij} + \lambda_j}{\sum_{j=1}^{J} \sum_{i=1}^{N} I(Y_i = k)x_{ij}}
\]
Naive Bayes and Unigram Language Models

The probability a new document has $\tau_{ik} = 1$ is then

$$p(\tau_{ik} = 1 | x_i, \hat{\pi}, \hat{\theta}) \propto p(\tau_{ik} = 1) p(x_i | \theta, \tau_{ik} = 1) \propto \hat{\pi}_k \prod_{j=1}^J (\hat{\theta}_{jk}) x_{ij} \propto p(C_k) \hat{\pi}_k \prod_{j=1}^J (\hat{\theta}_{jk}) x_{ij} \propto p(C_k) \hat{\pi}_k \prod_{j=1}^J (\hat{\theta}_{jk}) x_{ij} \propto p(C_k) \hat{\pi}_k \prod_{j=1}^J (\hat{\theta}_{jk}) x_{ij}$$
Naive Bayes and Unigram Language Models

The probability a new document has $\tau_{ik} = 1$ is then

$$p(\tau_{ik} = 1|x_i, \hat{\pi}, \hat{\theta}) \propto p(\tau_{ik} = 1)p(x_i|\theta, \tau_{ik} = 1)$$
Naive Bayes and Unigram Language Models

The probability a new document has $\tau_{ik} = 1$ is then

$$p(\tau_{ik} = 1|x_i, \hat{\pi}, \hat{\theta}) \propto p(\tau_{ik} = 1)p(x_i|\theta, \tau_{ik} = 1)$$

$$\propto \hat{\pi}_k \prod_{j=1}^{J} (\hat{\theta}_{jk})^{x_{ij}}$$
Naive Bayes and Unigram Language Models

The probability a new document has $\tau_{ik} = 1$ is then

$$
p(\tau_{ik} = 1 | x_i, \hat{\pi}, \hat{\theta}) \propto p(\tau_{ik} = 1) p(x_i | \theta, \tau_{ik} = 1)
$$

$$
\propto \hat{\pi}_k \prod_{j=1}^J \left( \hat{\theta}_{jk} \right)^{x_{ij}}
$$

$$
\propto \hat{\pi}_k \prod_{j=1}^J \left( \hat{\theta}_{jk} \right)^{x_{ij}}
$$

Unigram model
Some R Code

library(e1071)
dep<- c(labels, rep(NA, no.testSet))
dep<- as.factor(dep)
out<- naiveBayes(dep~., as.data.frame(tdm))
predicts<- predict(out, as.data.frame(tdm[-training.set,]))
Assessing Models (Elements of Statistical Learning)

- **Model Selection**: tuning parameters to select final model (next week’s discussion)
- **Model assessment**: after selecting model, estimating error in classification
Comparing Training and Validation Set

Text classification and model assessment
- Replicate classification exercise with validation set
- General principle of classification/prediction
- Compare supervised learning labels to hand labels

Confusion matrix
Comparing Training and Validation Set

Representation of Test Statistics from Dictionary week (along with some new ones)

<table>
<thead>
<tr>
<th>Classification (algorithm)</th>
<th>Actual Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>Liberal</td>
<td>True Liberal</td>
</tr>
<tr>
<td>Liberal</td>
<td>False Liberal</td>
</tr>
<tr>
<td>Conservative</td>
<td>False Conservative</td>
</tr>
<tr>
<td>Conservative</td>
<td>True Conservative</td>
</tr>
</tbody>
</table>
Comparing Training and Validation Set

Representation of Test Statistics from Dictionary week (along with some new ones)

<table>
<thead>
<tr>
<th>Classification (algorithm)</th>
<th>Actual Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>Liberal</td>
<td>True Liberal</td>
</tr>
<tr>
<td>Liberal</td>
<td>False Liberal</td>
</tr>
<tr>
<td>Conservative</td>
<td>False Conservative</td>
</tr>
<tr>
<td>Conservative</td>
<td>True Conservative</td>
</tr>
</tbody>
</table>

\[
\text{Accuracy} = \frac{\text{TrueLib} + \text{TrueCons}}{\text{TrueLib} + \text{TrueCons} + \text{FalseLib} + \text{FalseCons}}
\]
Comparing Training and Validation Set

Representation of Test Statistics from Dictionary week (along with some new ones)

<table>
<thead>
<tr>
<th>Classification (algorithm)</th>
<th>Actual Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>Liberal</td>
<td>True Liberal</td>
</tr>
<tr>
<td>Liberal</td>
<td>False Liberal</td>
</tr>
<tr>
<td>Conservative</td>
<td>False Conservative</td>
</tr>
<tr>
<td>Conservative</td>
<td>True Conservative</td>
</tr>
</tbody>
</table>

\[
\text{Accuracy} = \frac{\text{TrueLib} + \text{TrueCons}}{\text{TrueLib} + \text{TrueCons} + \text{FalseLib} + \text{FalseCons}}
\]

\[
\text{Precision}_{\text{Liberal}} = \frac{\text{TrueLib}}{\text{TrueLib} + \text{False Liberal}}
\]
Comparing Training and Validation Set

Representation of Test Statistics from Dictionary week (along with some new ones)

<table>
<thead>
<tr>
<th>Classification (algorithm)</th>
<th>Actual Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>Liberal</td>
<td>Liberal</td>
</tr>
<tr>
<td>Liberal</td>
<td>True Liberal</td>
</tr>
<tr>
<td>Liberal</td>
<td>False Liberal</td>
</tr>
<tr>
<td>Conservative</td>
<td>False Conservative</td>
</tr>
<tr>
<td>Conservative</td>
<td>True Conservative</td>
</tr>
</tbody>
</table>

\[
\text{Accuracy} = \frac{\text{TrueLib} + \text{TrueCons}}{	ext{TrueLib} + \text{TrueCons} + \text{FalseLib} + \text{FalseCons}}
\]

\[
\text{Precision}_{\text{Liberal}} = \frac{\text{True Liberal}}{\text{True Liberal} + \text{False Liberal}}
\]

\[
\text{Recall}_{\text{Liberal}} = \frac{\text{True Liberal}}{\text{True Liberal} + \text{False Conservative}}
\]
Comparing Training and Validation Set

Representation of Test Statistics from Dictionary week (along with some new ones)

<table>
<thead>
<tr>
<th>Classification (algorithm)</th>
<th>Actual Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>Liberal</td>
<td>Liberal</td>
</tr>
<tr>
<td></td>
<td>True Liberal</td>
</tr>
<tr>
<td></td>
<td>False Liberal</td>
</tr>
<tr>
<td>Conservative</td>
<td>False Conservative</td>
</tr>
<tr>
<td></td>
<td>True Conservative</td>
</tr>
</tbody>
</table>

Accuracy: \[
\text{Accuracy} = \frac{\text{TrueLib} + \text{TrueCons}}{\text{TrueLib} + \text{TrueCons} + \text{FalseLib} + \text{FalseCons}}
\]

Precision_{Liberal}: \[
\text{Precision}_{\text{Liberal}} = \frac{\text{True Liberal}}{\text{True Liberal} + \text{False Liberal}}
\]

Recall_{Liberal}: \[
\text{Recall}_{\text{Liberal}} = \frac{\text{True Liberal}}{\text{True Liberal} + \text{False Conservative}}
\]

\(F_{\text{Liberal}}\): \[
F_{\text{Liberal}} = \frac{2\text{Precision}_{\text{Liberal}} \times \text{Recall}_{\text{Liberal}}}{\text{Precision}_{\text{Liberal}} + \text{Recall}_{\text{Liberal}}}
\]
Comparing Training and Validation Set

Representation of Test Statistics from Dictionary week (along with some new ones)

<table>
<thead>
<tr>
<th>Classification (algorithm)</th>
<th>Actual Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>Liberal</td>
<td>True Liberal</td>
</tr>
<tr>
<td></td>
<td>False Liberal</td>
</tr>
<tr>
<td>Conservative</td>
<td>False Conservative</td>
</tr>
<tr>
<td></td>
<td>True Conservative</td>
</tr>
</tbody>
</table>

Accuracy = \frac{\text{TrueLib} + \text{TrueCons}}{\text{TrueLib} + \text{TrueCons} + \text{FalseLib} + \text{FalseCons}}

Precision_{\text{Liberal}} = \frac{\text{True Liberal}}{\text{True Liberal} + \text{False Liberal}}

Recall_{\text{Liberal}} = \frac{\text{True Liberal}}{\text{True Liberal} + \text{False Conservative}}

F_{\text{Liberal}} = \frac{2 \cdot \text{Precision}_{\text{Liberal}} \cdot \text{Recall}_{\text{Liberal}}}{\text{Precision}_{\text{Liberal}} + \text{Recall}_{\text{Liberal}}}

Justin Grimmer (Stanford University)  
Text as Data  
November 6th, 2014  
29 / 34
ROC Curve

ROC as a measure of model performance

\[
\text{Recall}_{\text{Liberal}} = \frac{\text{True Liberal}}{\text{True Liberal} + \text{False Conservative}}
\]

\[
\text{Recall}_{\text{Conservative}} = \frac{\text{True Conservative}}{\text{True Conservative} + \text{False Liberal}}
\]

Tension:
- Everything liberal: \( \text{Recall}_{\text{Liberal}} = 1 \); \( \text{Recall}_{\text{Conservative}} = 0 \)
- Everything conservative: \( \text{Recall}_{\text{Liberal}} = 0 \); \( \text{Recall}_{\text{Conservative}} = 1 \)

Characterize Tradeoff:
Plot True Positive Rate \( \text{Recall}_{\text{Liberal}} \)
False Positive Rate \( (1 - \text{Recall}_{\text{Conservative}}) \)
Precision/Recall Tradeoff
Simple Classification Example

Analyzing house press releases

**Hand Code:** 1,000 press releases

- Advertising
- Credit Claiming
- Position Taking

Divide 1,000 press releases into two sets

- 500: Training set
- 500: Test set

**Initial exploration:** provides baseline measurement at classifier performances

**Improve:** through improving model fit
## Example from Ongoing Work

<table>
<thead>
<tr>
<th>Classification (Naive Bayes)</th>
<th>Actual Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>Position Taking</td>
<td>Advertising</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>40</td>
</tr>
<tr>
<td>80</td>
<td>60</td>
</tr>
</tbody>
</table>

Accuracy = \(\frac{10 + 40 + 306}{500} = 0.71\)

Precision\(_{PT}\) = \(\frac{10}{10} = 1\)

Recall\(_{PT}\) = \(\frac{10}{10 + 2 + 80} = 0.11\)

Precision\(_{AD}\) = \(\frac{40}{40 + 2 + 2} = 0.91\)

Recall\(_{AD}\) = \(\frac{40}{40 + 60} = 0.4\)

Precision\(_{Credit}\) = \(\frac{306}{306 + 80 + 60} = 0.67\)

Recall\(_{Credit}\) = \(\frac{306}{306 + 2} = 0.99\)
**Fit Statistics in R**

**RWeka** library provides *Amazing* functionality.

We’ll have more to say on how to install, use this next week!