Introduction to Natural Language Processing
This week: More on Classification
POS tags as classification
Lexical Semantics
Homework 4
1.4 Sentiment Classification Competition PART 2

For this next part of the assignment, you must implement LIWC features. In Assignment 3, you were asked to report the relative frequency of each feature. For this assignment, we want you to report the value of each feature using "binning". There is an example of how to do this in the add_lexical_features method in classify_movies.py.

As noted before, for every combination of features you consider, you should train it on both the Naive Bayes classifier and the Decision Tree classifier. For DecisionTree, just report the accuracy. For NaiveBayes, once you have determined the number of features that returns the highest accuracy, you should report that accuracy and the number of features that produced it. You should create a table containing all this information, along with some indication of what the feature set included. An example results table can be seen in Table 1.

<table>
<thead>
<tr>
<th>Feature Set</th>
<th>Naive Bayes</th>
<th>Decision Tree</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unigrams</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>Bigrams</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>Unigrams + LIWC</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>Unigrams + bigrams + POS-uni + POS-bi + LIWC</td>
<td>0.5</td>
<td>0.5</td>
</tr>
</tbody>
</table>

Table 1: Example Results Table
2 Labov & Waletzky Classification

2.1 Overview

In class we discussed the Labov & Waletzky theory of oral narrative analysis and presented the classification scheme of classifying narrative utterances into action, orientation and evaluation. Action utterances describe events in the story, while orientation utterances provide background or non-changing information about the story setting and characters, and evaluation utterances provide information on the emotions and thoughts of the narrator or story characters.

2.2 The Data

Your data are the following six files: training, development and test for Aesop’s Fables and for the story blogs. As before, we have a held out test set that we will test your program on.

Each dataset contains clauses from narratives. We are giving these to you as CSV files named fables-train.csv, blogs-train.csv, etc. Each line of the corresponding CSV file has the following format:

```
STORY#, CATEGORY, CLAUSE
STORY#, CATEGORY, CLAUSE
....
```
What is this Labov & Waletzky stuff?
And why are we doing it?
A brief discursion
Real World Data & Problems
vs.
Textbook data and problems
Diagram of supervised classification

(a) Training
- Input
- Feature extractor
- Features
- Machine learning algorithm
- Label

(b) Prediction
- Input
- Feature extractor
- Features
- Classifier model
- Label

http://www.nltk.org/howto/classify.html
Diagram of supervised classification

- What is the INPUT?
- What is the LABEL?
Diagram of supervised classification

(a) Training
- Input
- Feature extractor
- Features
- Label

(b) Prediction
- Input
- Feature extractor
- Features

- What could be the features?
- What do we already know how to extract from texts?
- What other possibilities might there be?
Diagram of supervised classification

- What is a machine learning algorithm?
- **It's just a function.** Takes features as input and returns an output.
- Simplest example: Linear regression
  - $Y = Mx + B$
  - $Y = w_1x_1 + w_2x_2 + ... w_nx_n + B$
- What format does the machine learning algorithm want?
  - A **VECTOR** of features and a **LABEL**
Classify Clause Types in Narrative Texts

Where do we get the labels from?
Where do we get the narrative texts?
Bill Labov Theory of Narrative

William Labov,
John H. and Margaret B. Fassitt Professor

http://nlds.soe.ucsc.edu
We only use a subset

Because our impression is that some of these are hard to label reliably

And we labelled this data ourselves
Clause Types in Oral Narrative

- Orientation
- Action
- Evaluation

A hungry Fox saw some fine bunches of Grapes hanging from a vine that was trained along a high trellis, and did his best to reach them by jumping as high as he could into the air. But it was all in vain, for they were just out of reach.
### Blogs: Orientation, Action, and Evaluation

Example of blog stories labeled by L&W categories

<table>
<thead>
<tr>
<th>#</th>
<th>Category</th>
<th>Story Clause</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Orientation</td>
<td>Now, on with this week’s story...</td>
</tr>
<tr>
<td>2</td>
<td>Orientation</td>
<td>The last month has been hectic.</td>
</tr>
<tr>
<td>3</td>
<td>Orientation</td>
<td>Turbo charged.</td>
</tr>
<tr>
<td>4</td>
<td>Orientation</td>
<td>Lot’s of work because I was learning from Tim, my partner in crime.</td>
</tr>
<tr>
<td>5</td>
<td>Orientation</td>
<td>This hasn’t been helped by the intense pressure in town due to the political transition coming to an end.</td>
</tr>
<tr>
<td>6</td>
<td>Orientation</td>
<td>This week things started alright and on schedule.</td>
</tr>
<tr>
<td>7</td>
<td>Action</td>
<td>But I managed to get myself arrested by the traffic police (roulage) early last Wednesday.</td>
</tr>
<tr>
<td>8</td>
<td>Action</td>
<td>After yelling excessively at their outright corrupted methods and asking incessantly for what law I actually broke,</td>
</tr>
<tr>
<td>9</td>
<td>Action</td>
<td>they managed to bring me in at the police HQ.</td>
</tr>
<tr>
<td>10</td>
<td>Action</td>
<td>I was drawing too much of a curious crowd for the authorities.</td>
</tr>
<tr>
<td>11</td>
<td>Action</td>
<td>In about half an hour at police HQ I had charmed everyone around.</td>
</tr>
<tr>
<td>12</td>
<td>Action</td>
<td>I had prepared my “gift” as they wished.</td>
</tr>
<tr>
<td>13</td>
<td>Evaluation</td>
<td>Decision withheld, they decided that I needn’t to bother,</td>
</tr>
<tr>
<td>14</td>
<td>Evaluation</td>
<td>they liked me too much.</td>
</tr>
<tr>
<td>15</td>
<td>Evaluation</td>
<td>I should go free.</td>
</tr>
<tr>
<td>16</td>
<td>Evaluation</td>
<td>I even managed to meet famous Raus, the big chief.</td>
</tr>
<tr>
<td>17</td>
<td>Evaluation</td>
<td>He was too happy to let me go when he realized I was no one.</td>
</tr>
<tr>
<td>18</td>
<td>Action</td>
<td>But then, a Major at his side noticed my Visa was expired.</td>
</tr>
<tr>
<td>19</td>
<td>Evaluation</td>
<td>Damn!</td>
</tr>
<tr>
<td>20</td>
<td>Orientation</td>
<td>My current Visa is being renewed in my other passport at Immigration’s.</td>
</tr>
<tr>
<td>21</td>
<td>Evaluation</td>
<td>Fuck.</td>
</tr>
<tr>
<td>22</td>
<td>Evaluation</td>
<td>In custody, for real.</td>
</tr>
</tbody>
</table>

Sequence of ACTIONS
Similarly for review classification
Sample Restaurant Review Website

Meson Sevilla

344 West 48th St
New York, NY 10036
Phon: (212)262-5590

Category: Spanish

Ratings average:
Food ❖❖❖❖❖
Service ❖❖❖❖❖
Price/Value ❖❖❖❖❖
Atmosphere ❖❖❖❖❖
Overall ❖❖❖❖❖

Number of reviews to date: 12
Paula Bacon (07/19/2005)
David Lindo (12/03/2004)
Tony Russo (07/28/2004)
Fred Morgan (07/18/2004)
Mike Russo (03/19/2004)

Add your review!
Email a friend
Sample Restaurant Review

Two of us shared the chorizo in red wine, marinated pork cubes, grilled calamari, artichokes and serrano ham and a tri-color salad. Every bite was wonderful, especially the chorizo. The list of wines by the glass is short, but the rioja was decent. If possible, order the tapas just 2 dishes at a time because they are best when they are fresh and hot (we just couldn't eat them fast enough). The bread was pretty unremarkable, but there was lots of it. I would definitely recommend this restaurant to anyone who likes a dining adventure. The ambience is charming and the service was friendly.

<table>
<thead>
<tr>
<th>Category</th>
<th>Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>Food</td>
<td>😊😊😊😊😊</td>
</tr>
<tr>
<td>Service</td>
<td>😊😊😊😊😊</td>
</tr>
<tr>
<td>Price/Value</td>
<td>😊😊😊😊😊</td>
</tr>
<tr>
<td>Atmosphere</td>
<td>😊😊😊😊😊</td>
</tr>
<tr>
<td>Overall</td>
<td>😊😊😊😊😊</td>
</tr>
</tbody>
</table>

- Was it a pleasant experience? Yes
- How many were seated at your table? 2
- Do they accept reservations? Yes
- Would you return? Yes
- If you ordered wine, what type was it? Rioja
- Credit cards accepted? Yes

Review submitted by: Paula Bacon (07/19/2005)
What kind of ‘correct labels’ are here?

Two of us shared the chorizo in red wine, marinated pork cubes, grilled calamari, artichokes and serrano ham and a tri-color salad. Every bite was wonderful, especially the chorizo. The list of wines by the glass is short, but the rioja was decent. If possible, order the tapas just 2 dishes at a time because they are best when they are fresh and hot (we just couldn't eat them fast enough). The bread was pretty unremarkable, but there was lots of it. I would definitely recommend this restaurant to anyone who likes a dining adventure. The ambience is charming and the service was friendly.

<table>
<thead>
<tr>
<th></th>
<th>Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>Food</td>
<td>☺️☺️☺️☺️☺️</td>
</tr>
<tr>
<td>Service</td>
<td>☺️☺️☺️☺️☺️</td>
</tr>
<tr>
<td>Price/Value</td>
<td>☺️☺️☺️☺️☺️</td>
</tr>
<tr>
<td>Atmosphere</td>
<td>☺️☺️☺️☺️☺️</td>
</tr>
<tr>
<td>Overall</td>
<td>☺️☺️☺️☺️☺️</td>
</tr>
<tr>
<td>Was it a pleasant experience?</td>
<td>Yes</td>
</tr>
<tr>
<td>How many were seated at your table?</td>
<td>2</td>
</tr>
<tr>
<td>Do they accept reservations?</td>
<td>Yes</td>
</tr>
<tr>
<td>Would you return?</td>
<td>Yes</td>
</tr>
<tr>
<td>If you ordered wine, what type was it?</td>
<td>Rioja</td>
</tr>
<tr>
<td>Credit cards accepted?</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Review submitted by: Paula Bacon (07/19/2005)

What can we use to predict them?
## Digital Camera User Reviews

**Nikon Coolpix 8700 Digital Camera**

**Overall rating:** ★★★★★

- Ease of Use: ★★★★★
- Durability: ★★★★★
- Battery Life: ★★★★★
- Photo Quality: ★★★★★
- Shutter Lag: ★★★★★

**Review Details:**

- **Compare Prices**
- **View Details**
- **Read Reviews**

### Write a Review

**Subscribe to reviews on this product**

### Read Reviews

<table>
<thead>
<tr>
<th>Sort by</th>
<th>Sort by</th>
<th>Page 1 2 - View all</th>
<th>Next</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Product Rating</strong></td>
<td><strong>Review Date</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Ease of Use:</strong></td>
<td>★★★★★</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Durability:</strong></td>
<td>★★★★★</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Battery Life:</strong></td>
<td>★★★★★</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Photo Quality:</strong></td>
<td>★★★★★</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Shutter Lag:</strong></td>
<td>★★★★★</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**The Coolpix 8700 Nikon joins the 8 megapixel club**

- **Pros:**
  - 8X Nikkor Zoom, 8 megapixel resolution
  - Con: Noisy images at ISO 400, AF hunts in low light, weak battery
- **Cons:**
  - Over the last couple of years Nikon has been losing market share to other manufacturers, but that negative sales trend may be due to change. Nikon's R&D folks have introduced two very exciting new digital cameras (the Coolpix 8700 and the D70 ... Read the full review

**Good Mid Level Digital Camera**

- **Pros:** Stylish, good quality photos, large memory capacity.
- **Cons:** Big and clunky, gets an error message a lot.

Let me state right away that I am not a camera professional or claim to be a techno guru. Far from it, I am a new mom who wanted to take loads of photos and share with family on the web. I know what I want out of a camera and that's about it. If you ...
How about here? What can we use?
File: 1 (17.html)
Review: 1
Atmosphere = 5
Do they accept reservations? = Yes
Food = 5
How many were seated at your table? = 2
Wine = "Red of some sort."
Overall = 5
Price/Value = 5
Service = 5
Text = "I think I may be the biggest cheerleader for the Tavolini anywhere, and here's why - ever since my wife and I started dating in high school, we've looked for somewhere to partake in that most-rare fine dining experience in Brantford, an otherwise terrible city to visit or live in. We first tried the Tavolini in the late 90's, and have returned several times, occasionally travelling halfway across the province to enjoy Anna's pasta, garlic bread and tartufo. We are treated like royalty every visit, even though we can never find a way to spend even a hundred dollars there - an amazing value for a full meal, with appetizers, drinks and desserts all around. The Tavolini is a truly special place, and if you have the misfortune to be passing through Brantford, you owe it to yourself to get off the highway, head downtown, lock the car up tight and enjoy some fine, fine food."
Was it a pleasant experience? = Yes
Would you return? = Yes
.
Review: 2
Atmosphere = 5
Do they accept reservations? = Yes
Food = 5
How many were seated at your table? = 4
Wine = "Valpolicella"
Overall = 5
Price/Value = 5
Service = 5
Text = "Best Italian food we ever had! Homemade fresh pasta and the veal dish my guest had was delicious! Very cozy, comfortable, homey atmosphere. The homemade tiramisu (the owner's grandmother's recipe) was out of this world! We will definitely be returning!"
Was it a pleasant experience? = Yes
Classification tasks

Assign the correct **class label** for a given input/object.

In basic classification tasks, each input is considered in isolation from all other inputs, and the set of labels is defined in advance.

Relevant Examples:

<table>
<thead>
<tr>
<th>Problem</th>
<th>Object</th>
<th>Label’s categories</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tagging</td>
<td>Word</td>
<td>Parts of speech</td>
</tr>
<tr>
<td>Dialogue Acts</td>
<td>Utterance</td>
<td>Greeting, Ask, Clarify</td>
</tr>
<tr>
<td>Task/Topic</td>
<td>Utterance</td>
<td>Directions, PhoneNum,</td>
</tr>
<tr>
<td>Review sentiment</td>
<td>Whole Review</td>
<td>POS, NEG</td>
</tr>
</tbody>
</table>

Adapted from: Foundations of Statistical NLP (Manning et al)

http://nlds.soe.ucsc.edu
Two ways to do classification

- Learn automatically from data how to assign utterances to classes
  - What we are doing

- Define patterns by hand that map to classes
  - The olden days
  - But sometimes still really useful
  - Can be used as a baseline against a trained classifier

- Want inspectable understandable models
Dialogue Acts in Facade

- Facade maps user natural language inputs into a set of dialogue acts, and responds to the act.

**NLU: Surface text to discourse acts**

Surface text: “You two look so happy in this wedding picture”

Discourse acts (~25):
- Agree
- Disagree
- Praise
- Refer to...

**Conversation management: Discourse acts to reactions**

- **Context:** Affinity Game
  - Proposer
  - Proposer
  - Priority map

- **Context:** Global
  - Proposer
  - Proposer
  - Priority map

Selector
End of discursion
1.4 Sentiment Classification Competition PART 2

For this next part of the assignment, you must implement LIWC features. In Assignment 3, you were asked to report the relative frequency of each feature. For this assignment, we want you to report the value of each feature using "binning". There is an example of how to do this in the add_lexical_features method in classify_movies.py.

As noted before, for every combination of features you consider, you should train it on both the Naive Bayes classifier and the Decision Tree classifier. For DecisionTree, just report the accuracy. For NaiveBayes, once you have determined the number of features that returns the highest accuracy, you should report that accuracy and the number of features that produced it. You should create a table containing all this information, along with some indication of what the feature set included. An example results table can be seen in Table 1.

<table>
<thead>
<tr>
<th>Feature Set</th>
<th>Naive Bayes</th>
<th>Decision Tree</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unigrams</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td># of features</td>
<td>512</td>
<td></td>
</tr>
<tr>
<td>Bigrams</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td># of features</td>
<td>64</td>
<td></td>
</tr>
<tr>
<td>Unigrams + LIWC</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td># of features</td>
<td>8192</td>
<td></td>
</tr>
<tr>
<td>Unigrams + bigrams +</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>POS-uni + POS-bi + LIWC</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td># of features</td>
<td>128</td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Example Results Table
2  Labov & Waletzky Classification

2.1  Overview

In class we discussed the Labov & Waletzky theory of oral narrative analysis and presented the classification scheme of classifying narrative utterances into ACTION, ORIENTATION and EVALUATION. Action utterances describe events in the story, while orientation utterances provide background or non-changing information about the story setting and characters, and evaluation utterances provide information on the emotions and thoughts of the narrator or story characters.

2.2  The Data

Your data are the following six files: training, development and test for Aesop’s Fables and for the story blogs. As before, we have a held out test set that we will test your program on.

Each dataset contains clauses from narratives. We are giving these to you as CSV files named fables-train.csv, blogs-train.csv, etc. Each line of the corresponding CSV file has the following format:

```
STORY#, CATEGORY, CLAUSE
STORY#, CATEGORY, CLAUSE
...
```

In CSV files

- Split by “,” get the labels, don’t need the story ID.
Things to Note

- The labels (action, orientation, evaluation)
- There are three vs. two.
- They are not evenly distributed in the data.
- Texts are much shorter than you have been dealing with. Reasonable to look at features generated and check them.
- Your feature extraction routines for the first part should work for this data, once you’ve split it and structured it so it's like the reviews data.
Story #:36 Category:Not Story Clause:

Story #:36 Category:Action Clause:Our first stop in Shanghai was the shopping mecca, Nanjing Lu for an altogether too familiar trip to Pearl City.

Story #:36 Category:Evaluation Clause:It was good for some haggling, gesturing, posturing, and finally a purchase.

Story #:36 Category:Evaluation Clause:Sadly (or not), it wouldn't be our last time in the Pearl Markets.

Story #:36 Category:Evaluation Clause:Nor our second to last (sadly).

Story #:36 Category:Action Clause:From there it was wandering time to the Yu Yuan gardens where we let the wiser members of the party escape the crowds of the bazaar into the gardens and awaited their return at Starbucks...

Story #:36 Category:Evaluation Clause:We've been immersed in the culture for long enough that we don't regard this as a sell-out anymore.

Story #:36 Category:Action Clause:More rambling took us to the riverfront, along the Bund, with a look across the Huang Pu River to the futuristic metallic ape of Pudong, including the Jin Mao Tower (which will soon be topped in height by it's neighbor) and the seemingly crayola inspired orbs of the Oriental Pearl TV Tower.

Story #:36 Category:Action Clause:We dallied as the dusk came and went and the lights began to decorate the night.

Story #:36 Category:Action Clause:Three of us would return in the morning to witness the health conscious out and about, including some troupes of Tai Chi aficionados, some of whom weren't afraid of a Marlboro to accompany the morning grind!
Using NLTK Dict type: see Chapter 5

- D1.update (D2) adds two dictionaries together

Python's Dictionary Methods: A summary of commonly-used methods and idioms involving dictionaries.

<table>
<thead>
<tr>
<th>Example</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>d = {}</td>
<td>create an empty dictionary and assign it to d</td>
</tr>
<tr>
<td>d[key] = value</td>
<td>assign a value to a given dictionary key</td>
</tr>
<tr>
<td>d.keys()</td>
<td>the list of keys of the dictionary</td>
</tr>
<tr>
<td>list(d)</td>
<td>the list of keys of the dictionary</td>
</tr>
<tr>
<td>sorted(d)</td>
<td>the keys of the dictionary, sorted</td>
</tr>
<tr>
<td>key in d</td>
<td>test whether a particular key is in the dictionary</td>
</tr>
<tr>
<td>for key in d</td>
<td>iterate over the keys of the dictionary</td>
</tr>
<tr>
<td>d.values()</td>
<td>the list of values in the dictionary</td>
</tr>
<tr>
<td>dict([(k1,v1), (k2,v2), ...])</td>
<td>create a dictionary from a list of key-value pairs</td>
</tr>
<tr>
<td>d1.update(d2)</td>
<td>add all items from d2 to d1</td>
</tr>
<tr>
<td>defaultdict(int)</td>
<td>a dictionary whose default value is zero</td>
</tr>
</tbody>
</table>
Finishing Classification today
Supervised vs. Unsupervised Classification

- A classifier is **supervised** if it is built on training corpora containing the correct label for each input.
  - This usually means that the program can calculate an error when the predicted label does not match the correct label.

- A classifier is **unsupervised** if it is built on training corpora that does not contain the correct label for each input.
  - There is no way to calculate an error.
  - Instead what we do is to see whether the classifications that we do help on another task that we do have labels for.
  - Example: pos and neg sentence classification helps with review classification?
Choosing the right features

- Use too few, and the data will be **underfitted**.
  - The classifier is too vague and makes too many mistakes.

- Use too many, and the data will be **overfitted**.
  - The classifier is too specific and will not generalize to new examples.
More on POS, NGRAMS, Language Models
The structure of an n-gram model

- The task can be re-stated in conditional probabilistic terms:
  \[ P(w_n \mid w_1 \ldots w_{n-1}) \]

- Limiting \( n \) under the Markov Assumption means:
  - greater chance of finding more than one occurrence of the sequence \( w_1 \ldots w_{n-1} \)
  - more robust statistical estimations

- N-grams are essentially equivalence classes or bins
  - every unique n-gram is a type or bin
  - \( BIN = A \ CHosen \ REPRESENTATION \ OF \ THE \ CONTEXT \)
P(A|B): Conditional Counts

- How can we use this information?
- a ConditionalFreqDist
- How often have I seen word$_2$ given that word$_1$ immediately preceded it?
- fox is seen exactly twice after having seen the

```python
>>> import nltk
>>> fables_text = open('cmps143/fables/TheFoxAndTheCrow.txt').read()
>>> sentences = nltk.sent_tokenize(fables_text)
>>> words = [nltk.word_tokenize(sentence) for sentence in sentences]
>>> flat_words = [word.lower() for sentence in words for word in sentence]
>>> bigrams = nltk.bigrams(flat_words)
>>> bgcdist = nltk.ConditionalFreqDist(bigrams)
>>> bgcdist.tabulate(conditions=["the", "fox", "and", "the", "crow"])
```

<table>
<thead>
<tr>
<th></th>
<th>birds</th>
<th>cheese</th>
<th>crow</th>
<th>fox</th>
<th>hue</th>
<th>just</th>
<th>observed</th>
<th>said</th>
<th>set</th>
<th>standing</th>
<th>that</th>
<th>the</th>
<th>tree</th>
<th>was</th>
</tr>
</thead>
<tbody>
<tr>
<td>the</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>fox</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>and</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>the</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>crow</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>
If we construct a model where all histories with the same n-1 words are considered one class or bin, we have an \((n-1)^{th}\) order Markov Model.

Note terminology:
- \textit{n-gram model} = \((n-1)^{th}\) order Markov Model
Size of n-gram models

- In a corpus of vocabulary size $N$, the assumption is that any combination of $n$ words is a potential $n$-gram.
- For a bigram model: $N^2$ possible $n$-grams in principle
- For a trigram model: $N^3$ possible n-grams.
- ...
Each n-gram in our model is a parameter used to estimate probability of the next possible word.
- too many parameters make the model unwieldy
- too many parameters lead to data sparseness: most of them will have $f = 0$ or $1$

Most models stick to unigrams, bigrams or trigrams.
- estimation can also combine different order models
Further considerations

- When building a model, we tend to take into account the start-of-sentence symbol:
  - *(the girl swallowed a large green caterpillar)*
  - *(<s> the)*
  - *(the girl)*
  - *(...)*

- Also typical to map all tokens $w$ such that $\text{count}(w) < k$ to <$\text{UNK}$>:
  - *(usually, tokens with frequency 1 or 2 are just considered “unknown” or “unseen”)*
  - *(this reduces the parameter space)*
Maximum Likelihood Estimation Approach

- Basic equation:
  \[ P(w_n \mid w_1...w_{n-1}) = \frac{P(w_1...w_n)}{P(w_1...w_{n-1})} \]

- \( P(A \mid B) \) is a ratio of the chances that both A and B happen, by the chances of B happening alone.
  \[ P(A \mid B) = \frac{P(A \cap B)}{P(B)} \]

- \( P(\text{ADJ} \mid \text{DET}) = P(\text{DET+ADJ}) / P(\text{DET}) \)

- In a unigram model, this reduces to simple probability.

- MLE models estimate probability using relative frequency.
What N should we use?

- Unigram models:
  - not entirely hopeless because most sentences contain a majority of highly common words
  - BUT ignore syntax completely:
    - \( P(\text{the tall man}) = P(\text{man the tall}) \)

- Bigrams: improve situation dramatically

- Trigram models great: **if you’ve seen it before!!**
  - Capture a surprising amount of contextual variation
  - Biggest limitation: **most** new trigrams in test data **will not have been seen** in training data.

- Even worse for 4-grams

- **NOTE:** Google makes available a pre-compiled 5 gram list

- Problem carries over to 4-grams, and is much worse!
Reliability vs. Discrimination

• larger n: more information about the context of the specific instance (greater discrimination)

• smaller n: more instances in training data, better statistical estimates (more reliability)
Example: Different ways of POS Tagging

POS tagging as a classification problem
Section 5.4 Tagging

- Tag everything  NN: 13% Accuracy!

```python
>>> tags = [tag for (word, tag) in brown.tagged_words(categories='news')]
>>> nltk.FreqDist(tags).max()
'NN'
```

Now we can create a tagger that tags everything as NN.

```python
>>> raw = 'I do not like green eggs and ham, I do not like them Sam I am!'
>>> tokens = word_tokenize(raw)
>>> default_tagger = nltk.DefaultTagger('NN')
>>> default_tagger.tag(tokens)
[('I', 'NN'), ('do', 'NN'), ('not', 'NN'), ('like', 'NN'), ('green', 'NN'), ('eggs', 'NN'), ('and', 'NN'), ('ham', 'NN'), ('Sam', 'NN'), ('I', 'NN'), ('am', 'NN'), ('!', 'NN')]
```

Unsurprisingly, this method performs rather poorly. On a typical corpus, it will tag only about an eighth of the tokens correctly, as we see below:

```python
>>> default_tagger.evaluate(brown_tagged_sents)
0.13089484257215028
```
RegExp Tagger: A rule-based classifier

- Use lists and morphology of the word (suffixes etc)
- Defaults to NN (last pattern)
- GETS 20.% !

```
>>> patterns = [
    (...)
    (r'\*\s$', 'NN'),  # nouns
    (r'^-[0-9]+([0-9]+)?$', 'CD'),  # cardinal numbers
    (r'\.*', 'NN')
    (...)
]
```

Note that these are processed in order, and the first one that matches is applied. Now we can set up a tagger and use it to tag a sentence. Now its right about a fifth of the time.

```
>>> regexp_tagger = nltk.RegexpTagger(patterns)
>>> regexp_tagger.tag(brown_sents[3])
[('````', 'NN'), ('Only', 'NN'), ('a', 'NN'), ('relative', 'NN'), ('handful', 'NN'), ('of', 'NN'), ('such', 'NN'), ('reports', 'NNS'), ('was', 'NNS'), ('received', 'VBD'), ('considering', 'VBG'), ('jury', 'NN'), ('said', 'NN'), ('widespread', 'NN')]
>>> regexp_tagger.evaluate(brown_tagged_sents)
0.20326391789486245
```
What do you think the unigram tagger does?
**Bigram and Trigram Taggers**

- The bigram tagger finds the most likely tag for each word, *given the preceding tag*. Gets 89.6%!!

- What is this backoff thing?

```python
>>> bigram_tagger = nltk.BigramTagger(brown_train, backoff=unigram_tagger_2)
>>> print bigram_tagger.size()
3379
>>> print 'Accuracy: %4.1f%%' % (...
   100.0 * bigram_tagger.evaluate(brown_test))
Accuracy: 89.6%
```

- Trigram Tagger: 89% !! *Why no better?*

```python
>>> trigram_tagger = nltk.TrigramTagger(brown_train, backoff=bigram_tagger)
>>> print trigram_tagger.size()
1495
>>> print 'Accuracy: %4.1f%%' % (...
   100.0 * trigram_tagger.evaluate(brown_test))
Accuracy: 89.0%
```