Admin

- Updated slides/examples on backoff with absolute discounting (I’ll review them again here today)
- Assignment 2
- Watson vs. Humans (tonight-Wednesday)

Backoff models: absolute discounting

\[ P_{\text{absolute}}(z \mid xy) = \begin{cases} \frac{C(xy) - D}{C(xy)} & \text{if } C(xy) > 0 \\ \alpha_{xy} P_{\text{absolute}}(z \mid y) & \text{otherwise} \end{cases} \]

- Subtract some absolute number from each of the counts (e.g. 0.75)
  - will have a large effect on low counts
  - will have a small effect on large counts

What is \( \alpha_{xy} \)?
### Backoff models: absolute discounting

<table>
<thead>
<tr>
<th>Event</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>see the dog</td>
<td>1</td>
</tr>
<tr>
<td>see the cat</td>
<td>2</td>
</tr>
<tr>
<td>see the banana</td>
<td>4</td>
</tr>
<tr>
<td>see the man</td>
<td>1</td>
</tr>
<tr>
<td>see the woman</td>
<td>1</td>
</tr>
<tr>
<td>see the car</td>
<td>1</td>
</tr>
</tbody>
</table>

### Backoff models: absolute discounting

\[
P_{\text{absolute}}(z | xy) = \begin{cases} 
C(xyz) - D & \text{if } C(xyz) > 0 \\
C(xy) & \text{otherwise}
\end{cases}
\]

\[
P_{\text{absolute}}(z | y) = \begin{cases} 
\alpha(xy) & \text{otherwise}
\end{cases}
\]

### Backoff models: absolute discounting

<table>
<thead>
<tr>
<th>Event</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>see the dog</td>
<td>1</td>
</tr>
<tr>
<td>see the cat</td>
<td>2</td>
</tr>
<tr>
<td>see the banana</td>
<td>4</td>
</tr>
<tr>
<td>see the man</td>
<td>1</td>
</tr>
<tr>
<td>see the woman</td>
<td>1</td>
</tr>
<tr>
<td>see the car</td>
<td>1</td>
</tr>
</tbody>
</table>

### Backoff models: absolute discounting

\[
P_{\text{absolute}}(z | xy) = \begin{cases} 
C(xyz) - D & \text{if } C(xyz) > 0 \\
C(xy) & \text{otherwise}
\end{cases}
\]

\[
P_{\text{absolute}}(z | y) = \begin{cases} 
\alpha(xy) & \text{otherwise}
\end{cases}
\]

### Backoff models: absolute discounting

<table>
<thead>
<tr>
<th>Event</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>see the dog</td>
<td>1</td>
</tr>
<tr>
<td>see the cat</td>
<td>2</td>
</tr>
<tr>
<td>see the banana</td>
<td>4</td>
</tr>
<tr>
<td>see the man</td>
<td>1</td>
</tr>
<tr>
<td>see the woman</td>
<td>1</td>
</tr>
<tr>
<td>see the car</td>
<td>1</td>
</tr>
</tbody>
</table>

α (see the) = ?

How much probability mass did we reserve/discount for the bigram model?

\[
P_{\text{absolute}}(z | xy) = \begin{cases} 
C(xyz) - D & \text{if } C(xyz) > 0 \\
C(xy) & \text{otherwise}
\end{cases}
\]

\[
P_{\text{absolute}}(z | y) = \begin{cases} 
\alpha(xy) & \text{otherwise}
\end{cases}
\]

p( puppy | see the ) = ?

p( rose | the Dow ) = ?

p( jumped | the Dow ) = ?

For each of the unique trigrams, we subtracted \(D/\text{count}(\text{"see the"})\) from the probability distribution.
### Backoff models: absolute discounting

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>see the dog</td>
<td>1</td>
</tr>
<tr>
<td>see the cat</td>
<td>2</td>
</tr>
<tr>
<td>see the banana</td>
<td>4</td>
</tr>
<tr>
<td>see the man</td>
<td>1</td>
</tr>
<tr>
<td>see the car</td>
<td>1</td>
</tr>
<tr>
<td>see the woman</td>
<td>1</td>
</tr>
</tbody>
</table>

\[ p(\text{puppy | see the}) = \alpha \]

\[ p(\text{see the}) = ? \]

\[ \alpha(\text{see the}) = ? \]

\[ \text{# of types starting with “see the”} \times D \]

\[ \text{count(“see the”) \times reserved mass(see the)} = \frac{6 \times 0.75}{10} = 0.45 \]

\[ \text{distribute this probability mass to all bigrams that we backed off to} \]

### Calculating \( \alpha \)

- We have some number of bigrams we’re going to backoff to, i.e. those \( X \) where \( \text{C(see the X)} = 0 \), that is unseen trigrams starting with “see the”.

- When we backoff, for each of these, we’ll be including their probability in the model: \( p(\text{X | the}) \)

- \( \alpha \) is the normalizing constant so that the sum of these probabilities equals the reserved probability mass:

\[ \sum_{X \text{such that } \text{C(see the X)}} p(\text{X | the}) = \text{reserved mass(see the)} \]

### Calculating \( \alpha \) in general: trigrams

- Calculate the reserved mass

\[ \text{reserved_mass(bigram)} = \frac{\text{# of types starting with bigram} \times D}{\text{count(bigram)}} \]

- Calculate the sum of the backed off probability. For bigram “A B”:

\[ 1 = \sum_{X \text{such that } \text{C(A B X)} > 0} p(X | B) \]

- Calculate \( \alpha \)

\[ \alpha(\text{A B}) = \frac{\text{reserved_mass(A B)}}{1 - \sum_{X \text{such that } \text{C(see the X)} > 0} p(\text{X | the})} \]
Calculating $\alpha$ in general: bigrams

- Calculate the reserved mass
  \[ \text{reserved_mass}(\text{unigram}) \equiv \frac{\# \text{ of types starting with unigram} \times D}{\text{count}(\text{unigram})} \]
- Calculate the sum of the backed off probability. For bigram "A B":
  \[ 1 - \sum_{X : C(AX) > 0} p(X) \]
  either is fine in practice, the left is easier
  \[ \sum_{X : C(AX) = 0} p(X) \]
- Calculate $\alpha$
  \[ \alpha(A) \equiv \frac{\text{reserved_mass}(A)}{1 - \sum_{X : C(AX) > 0} p(X)} \]
  $1 - \text{the sum of the unigram probabilities of those bigrams that we saw starting with word } A$

Calculating backoff models in practice

- Store the $\alpha$'s in another table
  - If it's a trigram backed off to a bigram, it's a table keyed by the bigrams
  - If it's a bigram backed off to a unigram, it's a table keyed by the unigrams
- Compute the $\alpha$'s during training
  - After calculating all of the probabilities of seen unigrams/bigrams/trigrams
  - Go back through and calculate the $\alpha$'s (you should have all of the information you need)
  - During testing, it should then be easy to apply the backoff model with the $\alpha$'s pre-calculated

Backoff models: absolute discounting

- $\text{reserved_mass} \equiv \frac{\# \text{ of types starting with bigram} \times D}{\text{count}(\text{bigram})}$
- Two nice attributes:
  - decreases if we've seen more bigrams
    should be more confident that the unseen trigram is no good
  - increases if the bigram tends to be followed by lots of other words
    will be more likely to see an unseen trigram

Syntactic structure

The man in the hat ran to the park.
Many possible CFGs for English, here is an example (fragment):
- $S \rightarrow NP \ VP$
- $VP \rightarrow V \ NP$
- $NP \rightarrow \text{DetP} \ N \mid \text{AdjP} \ NP$
- $\text{AdjP} \rightarrow \text{Adj} \mid \text{Adv AdjP}$
- $N \rightarrow \text{boy} \mid \text{girl}$
- $V \rightarrow \text{sees} \mid \text{likes}$
- $\text{Adj} \rightarrow \text{big} \mid \text{small}$
- $\text{Adv} \rightarrow \text{very}$
- $\text{DetP} \rightarrow \text{a} \mid \text{the}$

Grammar questions:
- Can we determine if a sentence is grammatical?
- Given a sentence, can we determine the syntactic structure?
- Can we determine how likely a sentence is to be grammatical? to be an English sentence?
- Can we generate candidate, grammatical sentences?

Parsing:
- Parsing is the field of NLP interested in automatically determining the syntactic structure of a sentence.
- Parsing can also be thought of as determining what sentences are “valid” English sentences.

Given a CFG and a sentence, determine the possible parse tree(s)

- $S \rightarrow NP \ VP$
- $NP \rightarrow \text{PP}$
- $NP \rightarrow N \ PP$
- $V \rightarrow V \ NP$
- $V \rightarrow V \ NP \ PP$
- $PP \rightarrow \text{IN} \ N$
- $\text{PP} \rightarrow \text{IN}$
- $V \rightarrow \text{eat}$
- $N \rightarrow \text{sushi}$
- $N \rightarrow \text{tuna}$
- $\text{IN} \rightarrow \text{with}$

I eat sushi with tuna

What parse trees are possible for this sentence? How did you figure it out?
Given a CFG and a sentence, determine the possible parse tree(s).

I eat sushi with tuna

approaches? algorithms?

What is the difference between these parses?

Top-down parsing
- start at the top (usually S) and apply rules
- matching left-hand sides and replacing with right-hand sides

Bottom-up parsing
- start at the bottom (i.e., words) and build the parse tree up from there
- matching right-hand sides and replacing with left-hand sides
Top Down Parsing

S
   /\  
  NP   VP
     \   
      Pronoun

Top Down Parsing

S
   /\  
  NP   VP
     \   
      X
        X
     book

Top Down Parsing

S
   /\  
  NP   VP
     \   
      ProperNoun

Top Down Parsing

S
   /\  
  NP   VP
     \   
      ProperNoun
        X
     book
Top Down Parsing

Top Down Parsing

Top Down Parsing

Top Down Parsing
Top Down Parsing

S
  ↓
 VP
  ↓
 Verb
  ↓
book

Top Down Parsing

S
  ↓
 VP
  ↓
 Verb
  ↓
book that
Top Down Parsing

```
S  
|   |      VP
|   |        
| Verb NP
```

Top Down Parsing

```
S  
|   |      VP
|   |        
| Verb NP
|   | book
```

Top Down Parsing

```
S  
|   |      VP
|   |        
| Verb NP
|   | book
| Pronoun
```

Top Down Parsing

```
S  
|   |      VP
|   |        
| Verb NP
|   | book
| Pronoun
|   | that
```

Top Down Parsing

$ S \rightarrow \text{VP}$

$\text{VP} \rightarrow \text{Verb} \text{ NP}$

book $\rightarrow \text{ProperNoun}$

Top Down Parsing

$ S \rightarrow \text{VP}$

$\text{VP} \rightarrow \text{Verb} \text{ NP}$

book $\rightarrow \text{ProperNoun}$

that

X

Top Down Parsing

$ S \rightarrow \text{VP}$

$\text{VP} \rightarrow \text{Verb} \text{ NP}$

book $\rightarrow \text{Det} \text{ Nominal}$

Top Down Parsing

$ S \rightarrow \text{VP}$

$\text{VP} \rightarrow \text{Verb} \text{ NP}$

book $\rightarrow \text{Det} \text{ Nominal}$

that
Top Down Parsing

S
  VP
    Verb NP
      book Det Nominal
        that Noun

Bottom Up Parsing

book that flight

Bottom Up Parsing

Noun
  book that flight
Bottom Up Parsing

Nominal
  Noun
    book
  that
    flight

Bottom Up Parsing

Nominal
  Nominal
    Noun
      book
    that
      flight

Bottom Up Parsing

Nominal
  Nominal
    Noun
      x
    flight

Bottom Up Parsing

Nominal
  Nominal
    PP
      book
    that
      flight
Bottom Up Parsing

Bottom Up Parsing

Bottom Up Parsing

Bottom Up Parsing
Bottom Up Parsing

```
VP
  VP
    Verb
    book
  PP
    Det
    that
    NP
    Nominal
    flight
```

```
VP
  NP
    Det
    that
    NP
    Nominal
    flight
```

```
S
  VP
    Verb
    book
    NP
    Nominal
    flight
```

```
S
  VP
    Verb
    book
    Det
    that
    NP
    Nominal
    flight
```
Parsing

- Pros/Cons?
  - Top-down:
    - Only examines parses that could be valid parses (i.e., with an S on top)
    - Doesn’t take into account the actual words!
  - Bottom-up:
    - Only examines structures that have the actual words as the leaves
    - Examines sub-parses that may not result in a valid parse!

Why is parsing hard?

- Actual grammars are large
- Lots of ambiguity!
  - Most sentences have many parses
  - Some sentences have a lot of parses
  - Even for sentences that are not ambiguous, there is often ambiguity for subtrees (i.e., multiple ways to parse a phrase)

Why is parsing hard?

I saw the man on the hill with the telescope

What are some interpretations?

Structural Ambiguity Can Give Exponential Parses

"I was on the hill that has a telescope when I saw a man."

"I saw a man who was on the hill that has a telescope on it."

"I was on the hill when I used the telescope to see a man."

"Using a telescope, I saw a man who was on a hill."

...
Dynamic Programming Parsing

- To avoid extensive repeated work you must cache intermediate results, specifically found constituents.
- Caching (memoizing) is critical to obtaining a polynomial time parsing (recognition) algorithm for CFGs.
- Dynamic programming algorithms based on both top-down and bottom-up search can achieve \(O(n^3)\) recognition time where \(n\) is the length of the input string.

Dynamic Programming Parsing Methods

- **CKY** (Cocke-Kasami-Younger) algorithm based on bottom-up parsing and requires first normalizing the grammar.
- **Earley parser** is based on top-down parsing and does not require normalizing grammar but is more complex.
- These both fall under the general category of **chart parsers** which retain completed constituents in a chart.

CKY

- First grammar must be converted to **Chomsky normal form (CNF)** in which productions must have either exactly 2 non-terminal symbols on the RHS or 1 terminal symbol (lexicon rules).
- Parse bottom-up storing phrases formed from all substrings in a triangular table (chart).

CNF Grammar

\[
S \rightarrow VP \\
VP \rightarrow VB NP \\
VP \rightarrow VB NP PP \\
NP \rightarrow DT NN \\
NP \rightarrow NN \\
NP \rightarrow NP PP \\
PP \rightarrow IN NP \\
IN \rightarrow with \\
VB \rightarrow film \\
VB \rightarrow trust \\
NN \rightarrow man \\
NN \rightarrow film \\
NN \rightarrow trust
\]