Assignment 2 out
- bigram language modeling
- Java
- Can work with partners
  - Anyone looking for a partner?
- 2a: Due Thursday
- 2b: Due Tuesday
- Style/commenting (JavaDoc)
- Some advice
  - Start now!
  - Spend 1-2 hours working out an example by hand (you can check your answers with me)
  - HashMap

Assignment submission: submit on-time!

Our first quiz (when?)
- In-class (~30 min.)
- Topics
  - corpus analysis
  - regular expressions
  - probability
  - language modeling
- Open book/notes
  - we’ll try it out for this one
  - better to assume closed book (30 minutes goes by fast!)
- 7.5% of your grade
Admin

Lab next class

Meet in Edmunds 105, 2:45-4pm

Today

Take home ideas:
- Key idea of smoothing is to redistribute the probability to handle less seen (or never seen) events
- Still must always maintain a true probability distribution
- Lots of ways of smoothing data
- Should take into account features in your data!

Smoothing

What if our test set contains the following sentence, but one of the trigrams never occurred in our training data?

\[
P(\text{I think today is a good day to be me}) =
\]
\[
P(\text{I}) \cdot P(\text{<start> | <start>}) \cdot x
\]

\[
P(\text{today | <start>}) \cdot x
\]

\[
P(\text{think}) \cdot x
\]

\[
P(\text{I think today}) \cdot x
\]

\[
P(\text{a | today is}) \cdot x
\]

\[
P(\text{good | is a}) \cdot x
\]

If any of these has never been seen before, prob = 0!
Smoothing

\[ P(\text{I think today is a good day to be me}) = \]
\[ P(\text{I | <start> <start>}) \times P(\text{think | <start> I}) \times P(\text{today | I think}) \times P(\text{is | think today}) \times P(\text{a | today is}) \times P(\text{good | is a}) \times \ldots \]

These probability estimates may be inaccurate. Smoothing can help reduce some of the noise.

The general smoothing problem

<table>
<thead>
<tr>
<th>modification</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>see the abacus</td>
<td>1</td>
</tr>
<tr>
<td>see the abbot</td>
<td>0</td>
</tr>
<tr>
<td>see the abduct</td>
<td>0</td>
</tr>
<tr>
<td>see the above</td>
<td>2</td>
</tr>
<tr>
<td>see the Abram</td>
<td>0</td>
</tr>
<tr>
<td>see the above</td>
<td>2</td>
</tr>
<tr>
<td>see the Abram</td>
<td>0</td>
</tr>
<tr>
<td>see the zygote</td>
<td>0</td>
</tr>
<tr>
<td>Total</td>
<td>3</td>
</tr>
</tbody>
</table>

Add-lambda smoothing

A large dictionary makes novel events too probable.

\[
\text{add } \lambda = 0.01 \text{ to all counts}
\]

<table>
<thead>
<tr>
<th>modification</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>see the abacus</td>
<td>1</td>
</tr>
<tr>
<td>see the abbot</td>
<td>0</td>
</tr>
<tr>
<td>see the abduct</td>
<td>0</td>
</tr>
<tr>
<td>see the above</td>
<td>2</td>
</tr>
<tr>
<td>see the Abram</td>
<td>0</td>
</tr>
<tr>
<td>see the zygote</td>
<td>0</td>
</tr>
<tr>
<td>Total</td>
<td>3</td>
</tr>
</tbody>
</table>

Add-lambda smoothing

How should we pick lambda?

<table>
<thead>
<tr>
<th>modification</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>see the abacus</td>
<td>1</td>
</tr>
<tr>
<td>see the abbot</td>
<td>0</td>
</tr>
<tr>
<td>see the abduct</td>
<td>0</td>
</tr>
<tr>
<td>see the above</td>
<td>2</td>
</tr>
<tr>
<td>see the Abram</td>
<td>0</td>
</tr>
<tr>
<td>see the zygote</td>
<td>0</td>
</tr>
<tr>
<td>Total</td>
<td>3</td>
</tr>
</tbody>
</table>
Setting smoothing parameters

Idea 1: try many $\lambda$ values & report the one that gets the best results?

Training | Test

Is this fair/appropriate?

Setting smoothing parameters

Training | Dev. | Test

collect counts from 80% of the data

pick $\lambda$ that gets best results on 20%

Now use that $\lambda$ to get smoothed counts from all 100%

and report results of that final model on test data.

problems? ideas?

Vocabulary

n-gram language modeling assumes we have a fixed vocabulary

- why?

Whether implicit or explicit, an n-gram language model is defined over a finite, fixed vocabulary

What happens when we encounter a word not in our vocabulary (Out Of Vocabulary)?

- If we don’t do anything, prob = 0
- Smoothing doesn’t really help us with this!

Vocabulary

To make this explicit, smoothing helps us with…

all entries in our vocabulary

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>see the abacus</td>
<td>1</td>
<td>1.01</td>
</tr>
<tr>
<td>see the abbot</td>
<td>0</td>
<td>0.01</td>
</tr>
<tr>
<td>see the abduct</td>
<td>0</td>
<td>0.01</td>
</tr>
<tr>
<td>see the above</td>
<td>2</td>
<td>2.01</td>
</tr>
<tr>
<td>see the Abram</td>
<td>0</td>
<td>0.01</td>
</tr>
<tr>
<td>see the zygote</td>
<td>0</td>
<td>0.01</td>
</tr>
</tbody>
</table>
Choosing a vocabulary: ideas?
- Grab a list of English words from somewhere
- Use all of the words in your training data
- Use some of the words in your training data
  - for example, all those that occur more than k times

Benefits/drawbacks?
- Ideally your vocabulary should represent words you’re likely to see
- Too many words: end up washing out your probability estimates (and getting poor estimates)
- Too few: lots of out of vocabulary

No matter your chosen vocabulary, you’re still going to have out of vocabulary (OOV)

How can we deal with this?
- Ignore words we’ve never seen before
  - Somewhat unsatisfying, though can work depending on the application
  - Probability is then dependent on how many in vocabulary words are seen in a sentence/text
- Use a special symbol for OOV words and estimate the probability of out of vocabulary

Add an extra word in your vocabulary to denote OOV (<OOV>, <UNK>)

Replace all words in your training corpus not in the vocabulary with <UNK>
- You’ll get bigrams, trigrams, etc with <UNK>
  - $p(<\text{UNK}> | \text{“I am”})$
  - $p(\text{fast} | \text{“I <UNK>”})$

During testing, similarly replace all OOV with <UNK>
Choosing a vocabulary

A common approach (and the one we’ll use for the assignment):

- Replace the first occurrence of each word by <UNK> in a data set
- Estimate probabilities normally

Vocabulary then is all words that occurred two or more times

This also discounts all word counts by 1 and gives that probability mass to <UNK>

Storing the table

How are we storing this table?

Should we store all entries?

<table>
<thead>
<tr>
<th>Word</th>
<th>Count</th>
<th>Count/Total</th>
<th>Probability</th>
<th>Corrected Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>see the abacus</td>
<td>1</td>
<td>1/3</td>
<td>1.01</td>
<td>1.01/203</td>
</tr>
<tr>
<td>see the abbot</td>
<td>0</td>
<td>0/3</td>
<td>0.01</td>
<td>0.01/203</td>
</tr>
<tr>
<td>see the abduct</td>
<td>0</td>
<td>0/3</td>
<td>0.01</td>
<td>0.01/203</td>
</tr>
<tr>
<td>see the above</td>
<td>2</td>
<td>2/3</td>
<td>2.01</td>
<td>2.01/203</td>
</tr>
<tr>
<td>see the Abram</td>
<td>0</td>
<td>0/3</td>
<td>0.01</td>
<td>0.01/203</td>
</tr>
<tr>
<td>see the abduct</td>
<td>0</td>
<td>0/3</td>
<td>0.01</td>
<td>0.01/203</td>
</tr>
<tr>
<td>see the zygote</td>
<td>0</td>
<td>0/3</td>
<td>0.01</td>
<td>0.01/203</td>
</tr>
<tr>
<td>Total</td>
<td>3</td>
<td>3/3</td>
<td>203</td>
<td>203</td>
</tr>
</tbody>
</table>

Storing the table

Hashtable (e.g. HashMap)

- fast retrieval
- fairly good memory usage

Only store those entries of things we’ve seen

- for example, we don’t store $|V|^2$ trigrams

For trigrams we can:

- Store one hashtable with bigrams as keys
- Store a hashtable of hashtables (I’m recommending this)

Storing the table: add-lambda smoothing

For those we’ve seen before:

Unsmoothed (MLE)       add-lambda smoothing

$P(c \mid ab) = \frac{C(abc)}{C(ab)}$       $P(c \mid ab) = \frac{C(abc) + \lambda}{C(ab) + \lambda}$

<table>
<thead>
<tr>
<th>Word</th>
<th>Count</th>
<th>Count/Total</th>
<th>Probability</th>
<th>Corrected Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>see the abacus</td>
<td>1</td>
<td>1/3</td>
<td>1.01</td>
<td>1.01/203</td>
</tr>
<tr>
<td>see the abbot</td>
<td>0</td>
<td>0/3</td>
<td>0.01</td>
<td>0.01/203</td>
</tr>
<tr>
<td>see the abduct</td>
<td>0</td>
<td>0/3</td>
<td>0.01</td>
<td>0.01/203</td>
</tr>
<tr>
<td>see the above</td>
<td>2</td>
<td>2/3</td>
<td>2.01</td>
<td>2.01/203</td>
</tr>
<tr>
<td>see the Abram</td>
<td>0</td>
<td>0/3</td>
<td>0.01</td>
<td>0.01/203</td>
</tr>
<tr>
<td>see the abduct</td>
<td>0</td>
<td>0/3</td>
<td>0.01</td>
<td>0.01/203</td>
</tr>
<tr>
<td>see the zygote</td>
<td>0</td>
<td>0/3</td>
<td>0.01</td>
<td>0.01/203</td>
</tr>
<tr>
<td>Total</td>
<td>3</td>
<td>3/3</td>
<td>203</td>
<td>203</td>
</tr>
</tbody>
</table>

What value do we need here to make sure it stays a probability distribution?
Storing the table: add-lambda smoothing

For those we've seen before:

Unsmoothed (MLE)

\[ P(c|ab) = \frac{C(abc)}{C(ab)} \]

- \[ P(c|ab) = \frac{C(abc) + \lambda}{C(ab) + \lambda V} \]

| Word | C(abc) | C(ab) | P(c|ab) |
|------|--------|-------|---------|
| see the abacus | 1 | 1.01 | 1.02 |
| see the abduct | 0 | 0.01 | 0.02 |
| see the addict | 0 | 0.01 | 0.02 |
| see the addict | 2 | 2.01 | 2.02 |
| see the abduct | 0 | 0.01 | 0.02 |
| see the abduct | 0 | 0.01 | 0.02 |
| Total | 1.01 | 1.02 | 2.03 |

Problems with frequency based smoothing

The following bigrams have never been seen:

\[ p(X|San) \]
\[ p(X|ate) \]

Which would add-lambda pick as most likely?

Which would you pick?

Witten-Bell Discounting

Some words are more likely to be followed by new words

- Diego
- Francisco
- Luis
- Jose
- Marcos
- food
- apples
- bananas
- ate
- hamburgers
- a lot
- for two
- grapes
- ...
Witten-Bell Discounting

Probability mass is shifted around, depending on the context of words

If \( P(w_i \mid w_{i-1}, \ldots, w_{i-m}) = 0 \), then the smoothed probability \( P_{WB}(w_i \mid w_{i-1}, \ldots, w_{i-m}) \) is higher if the sequence \( w_{i-1}, \ldots, w_{i-m} \) occurs with many different words \( w_k \).

Problems with frequency based smoothing

The following trigrams have never been seen:

\[ p( \text{car} \mid \text{see the} ) \]
\[ p( \text{zygote} \mid \text{see the} ) \]
\[ p( \text{cumquat} \mid \text{see the} ) \]

Which would add-lambda pick as most likely?
Witten-Bell?

Which would you pick?

Better smoothing approaches

Utilize information in lower-order models

Interpolation

- Combine probabilities of lower-order models in some linear combination

Backoff

\[
P_c(x|y) = \begin{cases} 
\frac{C^n(x|y)}{C^n(x|y) + \mu C^n(y)} & \text{if } C^n(x|y) > k \\
\frac{C^n(x|y) + \mu}{C^n(x) + \mu} & \text{otherwise}
\end{cases}
\]

- Often \( k = 0 \) (or 1)
- Combine the probabilities by "backing off" to lower models only when we don’t have enough information

Smoothing: Simple Interpolation

\[
P(z \mid xy) = \lambda \frac{C(xy)}{C(x)} + \mu \frac{C(y)}{C(y)} + (1 - \lambda - \mu) \frac{C(z)}{C(*)}
\]

Trigram is very context specific, very noisy

Unigram is context-independent, smooth

Interpolate Trigram, Bigram, Unigram for best combination

How should we determine \( \lambda \) and \( \mu \)?
Smoothing: Finding parameter values

Just like we talked about before, split training data into training and development.

Try lots of different values for $\lambda$, $\mu$ on heldout data, pick best.

Two approaches for finding these efficiently:

- EM (expectation maximization)
- "Powell search" — see Numerical Recipes in C

Backoff models: absolute discounting

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

Subtract some absolute number from each of the counts (e.g. 0.75)

- How will this affect rare words?
- How will this affect common words?

Backoff models: absolute discounting

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

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

| Trigram model: $p(z|xy)$ (before discounting) | Trigram model: $p(z|xy)$ (after discounting) | Bigram model: $p(z|y)$
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>seen trigrams (xyz occurred)</td>
<td>seen trigrams (xyz occurred)</td>
<td>$p(z</td>
</tr>
</tbody>
</table>

#### Example

- **Trigram model**
  - $p(\text{cat} | \text{see the}) = \frac{C(\text{cat}, \text{see the}) - D}{C(\text{see the})}$
  - if $C(\text{cat}, \text{see the}) > 0$
  - otherwise $\alpha(\text{see the}) P_{\text{absolute}}(\text{cat} | \text{y})$

- **Bigram model**
  - $P_{\text{absolute}}(\text{cat} | \text{y}) = C(\text{cat}, \text{see the}) - D$ if $C(\text{cat}, \text{see the}) > 0$
  - otherwise $\alpha(\text{see the}) P_{\text{absolute}}(\text{cat} | \text{y})$

#### Example probabilities

- **Trigram model**
  - $p(\text{cat} | \text{see the}) = \frac{2 - 0.75}{10} = 0.125$

- **Bigram model**
  - $P_{\text{absolute}}(\text{cat} | \text{y}) = 2 - D$ if $C(\text{cat}, \text{see the}) > 0$
  - otherwise $\alpha(\text{see the}) P_{\text{absolute}}(\text{cat} | \text{y})$
**Backoff models: absolute discounting**

<table>
<thead>
<tr>
<th>Bigram</th>
<th>Count</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>

$p(\text{puppy} \mid \text{see the}) = ?$

$\alpha(\text{see the}) = ?$

For each of the unique trigrams, we subtracted $D / \text{count("see the")}$ from the probability distribution.

$P_{\text{absolute}}(z \mid xy) = C(xyz) - D C(xy)$ if $C(xyz) > 0$

Calculating $\alpha$

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

When we backoff, for each of these, we’ll be including their probability in the model: $P(X \mid \text{the})$.

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

$$\alpha(\text{see the}) \sum_{X: C(\text{see the } X) = 0} p(X \mid \text{the}) = \text{reserved mass}(\text{see the})$$

**Calculating $\alpha$**

We can calculate $\alpha$ two ways:

- Based on those we haven’t seen:
  $$\alpha(\text{see the}) = \frac{\text{reserved mass}(\text{see the})}{\sum_{X: C(\text{see the } X) = 0} p(X \mid \text{the})}$$

- Or, more often, based on those we do see:
  $$\alpha(\text{see the}) = \frac{\text{reserved mass}(\text{see the})}{1 - \sum_{X: C(\text{see the } X) > 0} p(X \mid \text{the})}$$
### 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":
  \[ 1 - \sum_{X \text{ such that } X \text{ is seen after } A\ B} p(X) \text{ either is fine in practice, the left is easier} \]

- **Calculate $\alpha$**
  \[ \alpha(A\ B) = \frac{\text{reserved\_mass(A\ B)}}{1 - \sum_{X \text{ such that } X \text{ is seen after } A\ B} p(X)} \]

### Calculating $\alpha$ in general: bigrams

- **Calculate the reserved mass**
  \[ \text{reserved\_mass(unigram)} = \frac{\text{# of types starting with unigram} \times D}{\text{count(unigram)}} \]

- **Calculate the sum of the backed off probability.** For unigram "A":
  \[ 1 - \sum_{X \text{ such that } X \text{ starts with } A} p(X) \text{ either is fine in practice, the left is easier} \]

- **Calculate $\alpha$**
  \[ \alpha(A) = \frac{\text{reserved\_mass(A)}}{1 - \sum_{X \text{ such that } X \text{ starts with } A} p(X)} \]

### 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

- **the Dow Jones** 10
- **the Dow rose** 5
- **the Dow fell** 5

What is the reserved mass?

\[ \text{count(} \text{"see the"}) \times D \]

\[ \text{reserved\_mass(see the)} = \frac{3 \times D}{20} = \frac{3 \times 0.75}{20} = 0.115 \]

\[ \alpha(\text{the Dow}) = \frac{\text{reserved\_mass(see the)}}{1 - \sum_{X \text{ such that } X \text{ is seen before } \text{the Dow}} p(X)} \]
Backoff models: absolute discounting

reserved_mass = \frac{\text{# of types starting with bigram} \cdot D}{\text{count(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