Admin

- How did assignment 1 go?
  - How did you feel about not handing in code?

- Assignment 2 will be out soon on language modeling

- Readings
  - make sure you’re keeping up with them
  - I will post a popular media article for next week (probably Monday) to read and discuss in class

In-class exercise

- How did it go?
  - Did you make it through all of the questions?

- Estimating probabilities
  - How accurate were your estimates for the average draw from 1-13 with 10, 50 and 100 draws?
  - How accurate were your estimates for the single card?

- Poker face
  - How probable is a royal flush? How does this compare to NLP probabilities?

- Birthdays
  - Any shared birthdays?
  - Anyone’s birthday that day? week?

- Monty hall
  - should you switch?

- The Coin game
  - HHH vs. THT
  - This is sort of like the language modeling task we’ll look at today
Independence

- Two variables are independent if they do not effect each other.

- For two independent variables, knowing the value of one does not change the probability distribution of the other variable.
  - The result of the toss of a coin is independent of a roll of a dice.
  - Price of tea in England is independent of the result of a general election in Canada.

Independent or Dependent?

- Catching a cold and enjoying reading books
- Miles per gallon and driving habits
- Height and longevity of life

Independent variables

- How does independence affect our probability equations/properties?

- If $A$ and $B$ are independent (written ...)
  - $P(A,B) = P(A)P(B)$
  - $P(A|B) = P(A)$
  - $P(B|A) = P(B)$

Conditional Independence

- Dependent events can become independent given certain other events.

- Examples,
  - Height and length of life
  - "Correlation" studies
  - Size of your lawn and length of life.

- If $A$, $B$ are conditionally independent of $C$
  - $P(A,B|C) = P(A|C)P(B|C)$
  - $P(A|B,C) = P(A|C)$
  - $P(B|A,C) = P(B|C)$
  - but $P(A,B) \neq P(A)P(B)$
Assume independence

- Sometimes we will assume two variables are independent (or conditionally independent) even though they’re not

- Why?
  - Creates a simpler model
  - $p(X,Y)$ many more variables than just $p(X)$ and $p(Y)$
  - May not be able to estimate the more complicated model

Language modeling

- What does natural language look like?

- More specifically in NLP, probabilistic model

- Two related questions:
  - $p(\text{sentence} )$
    - $p(\text{"I like to eat pizza"})$
    - $p(\text{"pizza like I eat"})$
  - $p(\text{word | previous words} )$
    - $p(\text{"pizza" | "I like to eat"})$
    - $p(\text{"garbage" | "I like to eat"})$
    - $p(\text{"run" | "I like to eat"})$

Language modeling

- How might these models be useful?
  - Language generation tasks
    - machine translation
    - summarization
    - simplification
    - speech recognition
    - …
  - Text correction
    - spelling correction
    - grammar correction

Ideas?

- $p(\text{"I like to eat pizza"})$
- $p(\text{"pizza like I eat"})$
- $p(\text{"pizza" | "I like to eat"})$
- $p(\text{"garbage" | "I like to eat"})$
- $p(\text{"run" | "I like to eat"})$
Look at a corpus

Language modeling

I think today is a good day to be me

Language modeling is about dealing with data sparsity!

Language model is really a probabilistic explanation of how the sentence was generated

Key idea:
- break this generation process into smaller steps
- estimate the probabilities of these smaller steps
- the overall probability is the combined product of the steps

Two approaches:
- n-gram language modeling
  - Start at the beginning of the sentence
  - Generate one word at a time based on the previous words

- syntax-based language modeling
  - Construct the syntactic tree from the top down
  - e.g. context free grammar
  - eventually at the leaves, generate the words

Pros/cons?
n-gram language modeling

I think today is a good day to be me

Google

Our friend the chain rule

Step 1: decompose the probability

\[ P(\text{I think today is a good day to be me}) = \]

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

How can we simplify these?

The n-gram approximation

Assume each word depends only on the previous n-1 words (e.g. trigram: three words total)

\[ P(\text{is | I think today}) = P(\text{is | think today}) \]

\[ P(\text{a | I think today is}) = P(\text{a | today is}) \]

\[ P(\text{good | I think today is a}) = P(\text{good | is a}) \]

Estimating probabilities

- How do we find probabilities? \( P(\text{is | think today}) \)
- Get real text, and start counting (MLE)!

\[ P(\text{is | think today}) = \frac{\text{count(\text{think today is})}}{\text{count(\text{think today})}} \]
Estimating from a corpus

Corpus of sentences
(e.g. gigaword corpus)

\[ \text{n-gram language model} \]

\[ \text{count all of the trigrams} \]

\[ \text{why do we need} \]
\[ \text{and} \]
\[ \text{why do we need} \]
\[ \text{and} \]
\[ \text{and} \]

Estimating from a corpus

I am a happy Pomona College student .

\[ \text{count all of the trigrams} \]

\[ \text{p}(c|a,b) = \frac{\text{count}(a,b,c)}{\text{count}(a,b)} \]

Estimating from a corpus

I am a happy Pomona College student .

\[ \text{count all of the bigrams} \]
Estimating from a corpus

1. Go through all sentences and count trigrams and bigrams
   - Usually you store these in some kind of data structure

2. Now, go through all of the trigrams and use the count and the bigram count to calculate MLE probabilities
   - Do we need to worry about divide by zero?

Applying a model

- Given a new sentence, we can apply the model

\[
\frac{p(\text{Pomona College students are the best .})}{p(\text{Pomona | <start> <start>}) \times p(\text{College | <start> Pomona}) \times p(\text{students | Pomona College}) \times \ldots \times p(\text{<end> | . <end>})}
\]

Some examples

Generating examples

- We can also use a trained model to generate a random sentence
- Ideas?

We have a distribution over all possible starting words

Draw one from this distribution

- p(A | <start> <start>)
- p(Apples | <start> <start>)
- p( | <start> <start>)
- p(The | <start> <start>)
- Zebra | <start> <start>)
Generating examples

- Unigram
  are were that éræs malel naturally built describes jazz territory heteromyids
  film tenor prime live founding must on was feet negro legal gate in an beside .
  provincial son ; stephenson simply spaces stretched performance double-entry
grove replacing station across to burmns . repairing éræs capital about double
reached omellos el time believed what hotels parameter jurisprudence wards
syndrome to éræs profanity is administrators éræs offices hilarous
institutionalized remains writer royalty client , éræs tyson , and objective .
instructions seem timekeeper has éræs valley éræs " magnitudes for love an éræs
from clifiklater , , , , éræs is belongs fame they the
corrected , , an in pressure %NUMBER% %NUMBER% her flavored éræs derogatory is von
macedon indirectly of crop duty leam earthbound éræs éræs dancing similarity
éræs named éræs berkely . , off-scale overtime , each manifold stripes dlnu
traffic ascertic and at alpha popularity town

- Bigrams
  the wikipedia county , mexico .
maurice ravel . it is require that is sparta , where functions . most
widely admired .
hologens chamiall cast jason against test site .

- Trigrams
  is widespread in north africa in june %NUMBER% %NUMBER% units were built by
  with .
  jewish video spiritual are considered irad , this season was an extratropical cyclone .
  the british railways ’ s strong and a spot .
We can train a language model on some data. How can we tell how well we're doing?

- For example:
  - Bigrams vs. trigrams
  - 100K sentence corpus vs. 100M
  - ...

A very good option: extrinsic evaluation.

- If you're going to be using it for machine translation, build a system with each language model and compare the two based on their approach for machine translation.

- Sometimes we don't know the application.
- Can be time consuming.

Common NLP/machine learning/AI approach:

- All sentences
  - Training sentences
  - Testing sentences

Test sentences

n-gram language model

Ideas?
Evaluation

- A good model should do a good job of predicting actual sentences

```
model 1
```

```
model 2
```

Perplexity

- View the problem as trying to predict the test corpus one word at a time in sequence
- A perfect model would always know give the next work probability 1

```
I like to eat banana peels.
```

Perplexity

- Perplexity is the average per-word probability

\[
\sqrt[n]{\prod_{i=1}^{n} P(w_i | w_{1..i-1})}
\]

- Sometimes is also written as

\[
\sqrt[n]{\prod_{i=1}^{n} P(w_i | w_{1..i-1})} = \frac{\sum_{i=1}^{n} \log P(w_i | w_{1..i-1})}{n}
\]

Another view of perplexity

- Weighted average branching factor
  - number of possible next words that can follow a word or phrase
  - measure of the complexity/uncertainty of text (as viewed from the language models perspective)
**Smoothing**

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

\[ P(I \text{ think today is a good day to be me}) = P(I|<\text{start}>}<\text{start}>x \]
\[ P(\text{think}|<\text{start}>x) \]
\[ P(\text{today}|\text{think}x) \quad \text{if any of these has never been seen before, prob } = 0! \]
\[ P(\text{is}|\text{think today})x \]
\[ P(\text{is}|\text{think today})x \]
\[ P(\text{good}|\text{is a})x \]
\[ \ldots \]

**A better approach**

- \( p(z|x) = ? \)
- Suppose our training data includes
  - \( x \ y \ a \ldots \)
  - \( x \ y \ d \ldots \)
  - \( x \ y \ d \ldots \)
  - but never: \( x \ y \ z \)
- We would conclude
  - \( p(a|x \ y) = 1/3? \)
  - \( p(d|x \ y) = 2/3? \)
  - \( p(z|x \ y) = 0/3? \)

- Is this ok?
- Intuitively, how should we fix these?

**Smoothing the estimates**

- **Basic idea:**
  - \( p(a|x \ y) = 1/3? \) reduce
  - \( p(d|x \ y) = 2/3? \) reduce
  - \( p(z|x \ y) = 0/3? \) increase

- **Discount** the positive counts somewhat
- **Reallocate** that probability to the zeroes

- **Remember,** it needs to stay a probability distribution

**Other situations**

- \( p(z|x \ y) = ? \)
- Suppose our training data includes
  - \( x \ y \ a \ldots \) (100 times)
  - \( x \ y \ d \ldots \) (100 times)
  - \( x \ y \ d \ldots \) (100 times)
  - but never: \( x \ y \ z \)
- Suppose our training data includes
  - \( x \ y \ a \ldots \)
  - \( x \ y \ d \ldots \)
  - \( x \ y \ d \ldots \)
  - \( x \ y \ldots \) (300 times)
  - but never: \( x \ y \ z \)

Is this the same situation as before?
Smoothing the estimates

- Should we conclude $p(a \mid xy) = 1/3$? **reduce**
- $p(d \mid xy) = 2/3$? **reduce**
- $p(z \mid xy) = 0/3$? **increase**

- Redoing the estimate is particularly important if:
  - the denominator is small …
    - $1/3$ probably too high, $100/300$ probably about right
  - numerator is small …
    - $1/300$ probably too high, $100/300$ probably about right

Add-one (Laplacian) smoothing

300 observations instead of 3 – better data, less smoothing

<table>
<thead>
<tr>
<th></th>
<th>xy</th>
<th>1</th>
<th>1/3</th>
<th>2</th>
<th>2/29</th>
</tr>
</thead>
<tbody>
<tr>
<td>xya</td>
<td>100</td>
<td>100/300</td>
<td>101</td>
<td>101/326</td>
<td></td>
</tr>
<tr>
<td>xyb</td>
<td>0</td>
<td>0/300</td>
<td>1</td>
<td>1/326</td>
<td></td>
</tr>
<tr>
<td>xyc</td>
<td>0</td>
<td>0/300</td>
<td>1</td>
<td>1/326</td>
<td></td>
</tr>
<tr>
<td>xyd</td>
<td>200</td>
<td>200/300</td>
<td>201</td>
<td>201/326</td>
<td></td>
</tr>
<tr>
<td>xye</td>
<td>0</td>
<td>0/300</td>
<td>1</td>
<td>1/326</td>
<td></td>
</tr>
<tr>
<td>xyz</td>
<td>0</td>
<td>0/300</td>
<td>1</td>
<td>1/326</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>300</td>
<td>300/300</td>
<td>326</td>
<td>326/326</td>
<td></td>
</tr>
</tbody>
</table>

Add-one (Laplacian) smoothing

What happens if we’re now considering 20,000 word types?

<table>
<thead>
<tr>
<th></th>
<th>xy</th>
<th>1</th>
<th>1/3</th>
<th>2</th>
<th>2/29</th>
</tr>
</thead>
<tbody>
<tr>
<td>xya</td>
<td>1</td>
<td>1/3</td>
<td>2</td>
<td>2/29</td>
<td></td>
</tr>
<tr>
<td>xyb</td>
<td>0</td>
<td>0/3</td>
<td>1</td>
<td>1/29</td>
<td></td>
</tr>
<tr>
<td>xyc</td>
<td>0</td>
<td>0/3</td>
<td>1</td>
<td>1/29</td>
<td></td>
</tr>
<tr>
<td>xyd</td>
<td>2</td>
<td>2/3</td>
<td>3</td>
<td>3/29</td>
<td></td>
</tr>
<tr>
<td>xye</td>
<td>0</td>
<td>0/3</td>
<td>1</td>
<td>1/29</td>
<td></td>
</tr>
<tr>
<td>xyz</td>
<td>0</td>
<td>0/3</td>
<td>1</td>
<td>1/29</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>3</td>
<td>3/3</td>
<td>29</td>
<td>29/29</td>
<td></td>
</tr>
</tbody>
</table>
Add-one (Laplacian) smoothing

20000 word types, not 26 letters

<table>
<thead>
<tr>
<th>event</th>
<th>count</th>
<th>probability</th>
<th>smoothing probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>see the abacus</td>
<td>1</td>
<td>1/3</td>
<td>2/20003</td>
</tr>
<tr>
<td>see the abbot</td>
<td>0</td>
<td>0/3</td>
<td>1/20003</td>
</tr>
<tr>
<td>see the abduct</td>
<td>0</td>
<td>0/3</td>
<td>1/20003</td>
</tr>
<tr>
<td>see the above</td>
<td>2</td>
<td>2/3</td>
<td>3/20003</td>
</tr>
<tr>
<td>see the Abram</td>
<td>0</td>
<td>0/3</td>
<td>1/20003</td>
</tr>
<tr>
<td>see the zygote</td>
<td>0</td>
<td>0/3</td>
<td>1/20003</td>
</tr>
<tr>
<td>Total</td>
<td>3</td>
<td>3/3</td>
<td>20003/20003</td>
</tr>
</tbody>
</table>

Any problem with this?

Add-lambda smoothing

A large dictionary makes novel events too probable.
Instead of adding 1 to all counts, add $\lambda = 0.01$?
This gives much less probability to novel events

<table>
<thead>
<tr>
<th>event</th>
<th>count</th>
<th>probability</th>
<th>smoothing probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>see the abacus</td>
<td>1</td>
<td>1/3</td>
<td>1.01/203</td>
</tr>
<tr>
<td>see the abbot</td>
<td>0</td>
<td>0/3</td>
<td>0.01/203</td>
</tr>
<tr>
<td>see the abduct</td>
<td>0</td>
<td>0/3</td>
<td>0.01/203</td>
</tr>
<tr>
<td>see the above</td>
<td>2</td>
<td>2/3</td>
<td>2.01/203</td>
</tr>
<tr>
<td>see the Abram</td>
<td>0</td>
<td>0/3</td>
<td>0.01/203</td>
</tr>
<tr>
<td>see the zygote</td>
<td>0</td>
<td>0/3</td>
<td>0.01/203</td>
</tr>
<tr>
<td>Total</td>
<td>3</td>
<td>3/3</td>
<td>203</td>
</tr>
</tbody>
</table>

The general smoothing problem

<table>
<thead>
<tr>
<th>event</th>
<th>count</th>
<th>probability</th>
<th>smoothing probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>see the abacus</td>
<td>1</td>
<td>1/3</td>
<td>?</td>
</tr>
<tr>
<td>see the abbot</td>
<td>0</td>
<td>0/3</td>
<td>?</td>
</tr>
<tr>
<td>see the abduct</td>
<td>0</td>
<td>0/3</td>
<td>?</td>
</tr>
<tr>
<td>see the above</td>
<td>2</td>
<td>2/3</td>
<td>?</td>
</tr>
<tr>
<td>see the Abram</td>
<td>0</td>
<td>0/3</td>
<td>?</td>
</tr>
<tr>
<td>see the zygote</td>
<td>0</td>
<td>0/3</td>
<td>?</td>
</tr>
<tr>
<td>Total</td>
<td>3</td>
<td>3/3</td>
<td>?</td>
</tr>
</tbody>
</table>
Add-lambda smoothing

How should we pick lambda?

<table>
<thead>
<tr>
<th>Category</th>
<th>Count</th>
<th>Fraction</th>
<th>Lambda</th>
<th>Result</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>...</td>
<td></td>
<td></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>0.01</td>
<td>0.01/203</td>
</tr>
</tbody>
</table>

Setting smoothing parameters

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

<table>
<thead>
<tr>
<th>Training</th>
<th>Test</th>
</tr>
</thead>
</table>

Is this fair/appropriate?

Correct experimentation

- General rules:
  - Test data should only be used for evaluation
  - No peeking! Only use it for your final results.
  - Never skew anything in your favor

- Other ideas?
Concerns

- 20% may not be enough to reliably determine \( \lambda \).
- We're maximizing lambda for only 80% of our data (will not be the same as the optimal for 100%)
- We're losing 20% of our data for calculating counts

Ideas?

Cross-validation (aka “jackknifing”)

- If 20% too little: try 5 training/test splits as below
  - Pick \( \lambda \) that gets best average performance
    
    ![Cross-validation Diagram]

  - This tests on all 100% (in turn), so we can more reliably assess \( \lambda \).
  - Unfortunately, still picks a \( \lambda \) that does well on 80% training.

N-fold Cross-Validation and “Leave One Out”

- Test each sentence with smoothed model from other N-1 sentences
  - Still tests on all 100% (in turn), so we can reliably assess \( \lambda \).
  - Tests if \( \lambda \) is good for smoothing \((N-1)/N = 100\%\) of training data, which matches our actual test conditions
  - Surprisingly fast: why?
    - Usually easy to change model by adding/subtracting 1 sentence’s counts