Independence

Two variables are independent if they do not affect 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 whether or not you get an A in NLP.

Independent or Dependent?

- You catching a cold and a butterfly flapping its wings in Africa.
- 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 $A \perp B$

- $P(A|B) = P(A)$
- $P(B|A) = P(B)$

What does that mean about $P(A,B)$?

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 $C$  

$A \perp B | C$

- $P(A,B | C) = P(A|C) P(B|C)$
- $P(A|B,C) = P(A|C)$
- $P(B|A,C) = P(B|C)$
- 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:
- $p(\text{sentence})$
  - $p(\text{"I like to eat pizza"})$
  - $p(\text{"pizza like I eat"})$
- $p(\text{word} | \text{previous words})$
  - $p(\text{"pizza"} | \text{"I like to eat"})$
  - $p(\text{"garbage"} | \text{"I like to eat"})$
  - $p(\text{"run"} | \text{"I like to eat"})$

Language modeling

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

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

Language modeling

I think today is a good day to be me

Probabilistic Language modeling

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

Language modeling

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

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? 

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

Estimating from a corpus

I am a happy Pomona College student.

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

\[ \text{<start> <start> I} \]
\[ \text{<start> I am} \]
\[ \text{I am a} \]
\[ \text{a happy Pomona} \]
\[ \text{happy Pomona College} \]
\[ \text{Pomona College student} \]
\[ \text{College student.} \]
\[ . <\text{end}> <\text{end}> \]

why do we need <\text{start}> and <\text{end}>?

Estimating from a corpus

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

\[ \text{<start> <start> I} \]
\[ \text{<start> I am} \]
\[ \text{I am a} \]
\[ \text{a happy Pomona} \]
\[ \text{happy Pomona College} \]
\[ \text{Pomona College student} \]
\[ \text{College student.} \]
\[ . <\text{end}> \]

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

Do we need to count anything else?
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:

\[
p(\text{Pomona College students are the best .}) = \prod
\]

\[
p(\text{Pomona} | \text{<start> <start>}) \times
p(\text{College} | \text{<start> Pomona}) \times
p(\text{students} | \text{Pomona College}) \times
\]

\[
\cdots
\]

\[
p(\text{<end>} | ., \text{<end>})
\]

Some examples

Generating examples

We can also use a trained model to generate a random sentence.

Ideas?

\[
\text{<start> <start>}
\]

We have a distribution over all possible starting words.

Draw one from this distribution.

\[
p(\text{A} | \text{<start> <start>})
\]

\[
p(\text{Apples} | \text{<start> <start>})
\]

\[
p(\text{I} | \text{<start> <start>})
\]

\[
p(\text{The} | \text{<start> <start>})
\]

\[
\cdots
\]

\[
p(\text{Zebras} | \text{<start> <start>})
\]
Generating examples

Unigram

are were that área mammal naturally built describes jazz territory heteromyids film tenor prime live founding must on was feet negro legal gate in on beside . provincial son ; stephenson simply spaces stretched performance double-entry grove replacing station across to burma , repairing áreas capital about double reached amrribus el time believed what hotels parameter jurisprudence words syndrome to áreas proficiency is administrators áreas offices hilarious institutionalized remains writer royalty dennis , áreas tyson , and objective , instructions seem timekeeper has áreas valley áreas ” magnitudes for love on áreas from allakaket , ” and central enlightened to , áreas is belongs fame they the corrected . on in pressure %NUMBER% her flavored áreas derogatory is won metcard indirectly of crop duty learn northbound áreas áreas dancing similarity áreas named áreas berkeley , off-scale overtime . each mansfield stripes dánu traffic aestetic and at alpha popularity town

Bigrams

the wikipedia county , mexico .

maurice ravel . it is require that is sparta , where functions . most widely admired .

halogens chamialli cast jason against test site .

Trigrams

is widespread in north africa in june %NUMBER% %NUMBER% %NUMBER% units were built by with .

jewish video spiritual are considered incd , this season was an extratropical cyclone .

the british railways ' s strong and a spot .
Evaluation

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
    - ...

Evaluation

A very good option: extrinsic evaluation

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

Sometimes we don’t know the application

Can be time consuming

Granularity of results

Evaluation

Common NLP/machine learning/AI approach

Training sentences

All sentences

Testing sentences

Evaluation

Test sentences

n-gram language model

Ideas?
Evaluation

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

Test sentences

<table>
<thead>
<tr>
<th>model 1</th>
<th>probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>model 2</td>
<td>probability</td>
</tr>
</tbody>
</table>

Pros: Fine for comparing two models
Cons: Doesn’t give us a sense of how well any model is doing

The problem

Which of these sentences will have a higher probability based on a language model?

I like to eat banana peels.

I like to eat banana peels with peanut butter.

Since probabilities are multiplicative (and between 0 and 1), they get smaller for longer sentences.
The solution: perplexity

\[ \text{prob}(w_{1,n}) = \prod_{i=1}^{n} p(w_i|w_{i-1}) \]

average the probabilities  
geometric mean

\[ PP(w_{1,n}) = \frac{1}{\sqrt[n]{\prod_{i=1}^{n} p(w_i|w_{i-1})}} \]

Calculating perplexity in practice

\[ \log \left( \frac{1}{\sqrt[n]{\prod_{i=1}^{n} p(w_i|w_{i-1})}} \right) = \log \left( \left( \frac{1}{\prod_{i=1}^{n} p(w_i|w_{i-1})} \right)^{1/n} \right) \]

\[ = \log \left( \frac{1}{\prod_{i=1}^{n} p(w_i|w_{i-1})} \right) \]

\[ = \frac{-\log(\prod_{i=1}^{n} p(w_i|w_{i-1}))}{n} \]

Average logprob per word!

Calculating perplexity in practice

\[ \log \left( \frac{1}{\sqrt[n]{\prod_{i=1}^{n} p(w_i|w_{i-1})}} \right) = \log \left( \left( \frac{1}{\prod_{i=1}^{n} p(w_i|w_{i-1})} \right)^{1/n} \right) \]

\[ = \frac{-\log(\prod_{i=1}^{n} p(w_i|w_{i-1}))}{n} \]

What is this?

- This is often how it’s calculated (and how we’ll calculate it)
- Avoid underflow from multiplying too many small probabilities together
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?

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

A better approach

\[
p(z | x y) = ? \]

Suppose our training data includes
\[
... x y a ... \]
\[
... x y d ... \]
\[
\text{but never: } xy z \]

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:
\[
p(a | x y) = 1/3? \quad \text{reduce} \]
\[
p(d | x y) = 2/3? \quad \text{reduce} \]
\[
p(z | x y) = 0/3? \quad \text{increase} \]

Discount the positive counts somewhat

Realocate that probability to the zeroes

Remember, it needs to stay a probability distribution
Other situations

\[ p(z \mid x y) = ? \]

Suppose our training data includes

... x y a ... (100 times)
... x y d ... (100 times)
but never: x y z

Suppose our training data includes

... x y a ...
... x y d ...
... x y d ...
... x y ... (300 times)
but never: x y z

Is this the same situation as before?

Smoothing the estimates

Should we conclude

\[ p(a \mid xy) = 1/3? \text{ reduce} \quad p(c \mid ab) = \text{count}(a b c) \]
\[ p(d \mid xy) = 2/3? \text{ reduce} \quad p(d \mid xy) = 0/3? \text{ increase} \quad \text{count}(a b) \]

Readjusting 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 is probably too high, 100/300 probably about right

Add-one (Laplacian) smoothing

<table>
<thead>
<tr>
<th></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>
</tr>
<tr>
<td>xyb</td>
<td>0</td>
<td>0/3</td>
<td>1</td>
<td>1/29</td>
</tr>
<tr>
<td>xyc</td>
<td>0</td>
<td>0/3</td>
<td>1</td>
<td>1/29</td>
</tr>
<tr>
<td>xyd</td>
<td>2</td>
<td>2/3</td>
<td>3</td>
<td>3/29</td>
</tr>
<tr>
<td>xye</td>
<td>0</td>
<td>0/3</td>
<td>1</td>
<td>1/29</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>xyz</td>
<td>0</td>
<td>0/3</td>
<td>1</td>
<td>1/29</td>
</tr>
<tr>
<td>Total xy</td>
<td>3</td>
<td>3/3</td>
<td>29</td>
<td>29/29</td>
</tr>
</tbody>
</table>

Add-one (Laplacian) smoothing

300 observations instead of 3 – better data, less smoothing

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

Total xy
Add-one (Laplacian) smoothing

What happens if we're now considering a vocabulary of 20,000 words?

<table>
<thead>
<tr>
<th></th>
<th>1/3</th>
<th>2/29</th>
</tr>
</thead>
<tbody>
<tr>
<td>xya</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>xyb</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>xyc</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>xyd</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>xye</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>xyz</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Total xy</td>
<td>3</td>
<td>29</td>
</tr>
</tbody>
</table>

Add-one (Laplacian) smoothing

20,000 words, not 26 letters

| see the abacus | 1 | 1/3 | 2 | 2/20003 |
| see the abbot | 0 | 0/3 | 1 | 1/20003 |
| see the abduct | 0 | 0/3 | 1 | 1/20003 |
| see the above | 2 | 2/3 | 3 | 3/20003 |
| see the Abram | 0 | 0/3 | 1 | 1/20003 |
| ... |     |     |   |        |
| see the zygote | 0 | 0/3 | 1 | 1/20003 |
| Total | 3 | 3/3 | 20003 | 20003/20003 |

Any problem with this?

Add-one (Laplacian) smoothing

An "unseen event" is a 0-count event

The probability of an unseen event is 19998/20003

The problem with add-one smoothing is it gives too much probability mass to unseen events

| see the abacus | 1 | 1/3 | 2 | 2/20003 |
| see the abbot | 0 | 0/3 | 1 | 1/20003 |
| see the abduct | 0 | 0/3 | 1 | 1/20003 |
| see the above | 2 | 2/3 | 3 | 3/20003 |
| see the Abram | 0 | 0/3 | 1 | 1/20003 |
| ... |     |     |   |        |
| see the zygote | 0 | 0/3 | 1 | 1/20003 |
| Total | 3 | 3/3 | 20003 | 20003/20003 |

The general smoothing problem

| see the abacus | 1 | 1/3 | ? | ? |
| see the abbot | 0 | 0/3 | ? | ? |
| see the abduct | 0 | 0/3 | ? | ? |
| see the above | 2 | 2/3 | ? | ? |
| see the Abram | 0 | 0/3 | ? | ? |
| ... |     |     |   |   |
| see the zygote | 0 | 0/3 | ? | ? |
| Total | 3 | 3/3 | ? | ? |
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></th>
<th>Count</th>
<th>Count/Total</th>
<th>Probability</th>
<th>Probability/Total</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 zygote</td>
<td>0</td>
<td>0/3</td>
<td>0.01</td>
<td>0.01/203</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>3</strong></td>
<td><strong>3/3</strong></td>
<td><strong>203</strong></td>
<td></td>
</tr>
</tbody>
</table>