LANGUAGE MODELING

David Kauchak
CS159 – Spring 2019

some slides adapted from Jason Eisner

Admin

How did assignment 1 go?

Assignment 2

Videos!

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 the whether or not you get an A in NLP

Independent or Dependent?

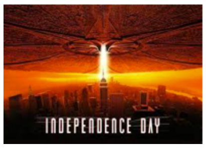
You catching a cold and a butterfly flapping its wings in Africa

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)?

Independent variables

How does independence affect our probability equations/properties?

If A and B are independent, written $A \perp B$

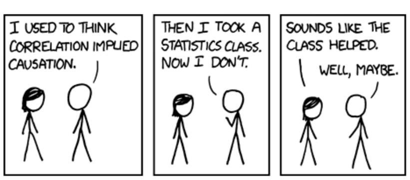
- P(A|B) = P(A)
- P(B|A) = P(B)
- P(A,B) = P(A|B) P(B) = P(A) P(B)
- P(A,B) = P(B|A) P(A) = P(A) P(B)

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



http://xkcd.com/552/

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) ≠ 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(sentence)
  - p("I like to eat pizza")
  - p("pizza like I eat")
- p(word | previous words)
  - p("pizza" | "I like to eat")
  - p("garbage" | "I like to eat")
  - p("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("I like to eat pizza")

p("pizza like I eat")

p("pizza" | "I like to eat")

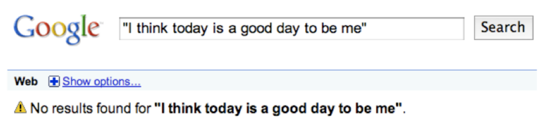
p("garbage" | "I like to eat")

p("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!

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

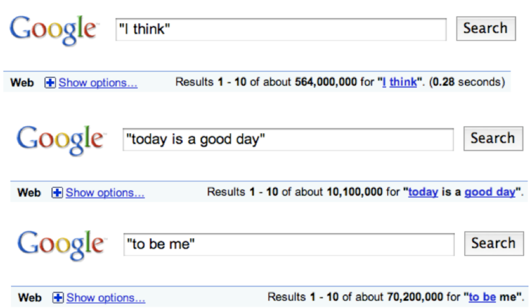
Two approaches:

- 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(I think today is a good day to be me) =

P(I | <start>) x

P(think | I) x

P(today | I think) x

P(is | I think today) x

P(a | I think today is) x

P(good | I think today is a) x

…

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(is | I think today) ≈ P(is | think today)

P(a | I think today is) ≈ P(a | today is)

P(good | I think today is a) ≈ P(good | is a)

Estimating probabilities

How do we find probabilities?

P(is | think today)

Get real text, and start counting (MLE)!

$$P(\text{is} \mid \text{think today}) = \frac{\text{count(think today is)}}{\text{count(think today)}}$$

Estimating from a corpus

Corpus of sentences
(e.g. gigaword corpus)

⋮

?

n-gram language model

Estimating from a corpus

I am a happy Pomona College student .

count all of the trigrams

<start> <start> I
<start> I am
I am a
am a happy
a happy Pomona
happy Pomona College
Pomona College student
College student .
student . <end>
. <end> <end>

why do we need <start> and <end>?

Estimating from a corpus

I am a happy Pomona College student .

count all of the trigrams

<start> <start> I
<start> I am
I am a
am a happy
a happy Pomona
happy Pomona College
Pomona College student
College student .
student . <end>
. <end> <end>

Do we need to count anything else?

Estimating from a corpus

I am a happy Pomona College student .

count all of the bigrams

<start> <start>
<start> I
I am
am a
a happy
happy Pomona
Pomona College
College student
student .
. <end>

$$p(c \mid a\ b) = \frac{\text{count}(a\ b\ c)}{\text{count}(a\ b)}$$

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(Pomona College students are the best .) = ?

p(Pomona | <start> <start>) *

p(College | <start> Pomona) *

p(students | Pomona College) *

⋮

p(<end> | . <end>) *

Some examples

Generating examples

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

Ideas?

<start> <start> ______

We have a distribution over all possible starting words

Draw one from this distribution

p(A | <start> <start>)

p(Apples | <start> <start>)

p(I | <start> <start>)

p(The | <start> <start>)

⋮

p(Zebras | <start> <start>)

Generating examples

<start> <start> Zebras ______

repeat!

p(are | <start> Zebras)

p(eat | <start> Zebras)

p(think | <start> Zebras)

p(and | <start> Zebras)

⋮

p(mostly | <start> Zebras)

Generation examples

Unigram

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

Generation examples

Bigrams

the wikipedia county , mexico .

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

halogens chamiali cast jason against test site .

Generation examples

Trigrams

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

jewish video spiritual are considered ircd , 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

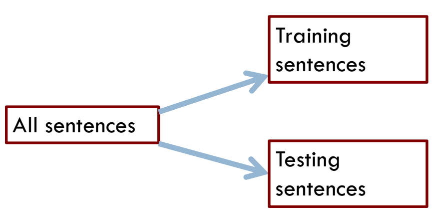
Sometimes we don't know the application

Can be time consuming

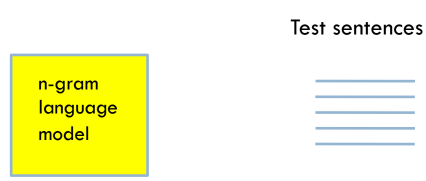
Granularity of results

Evaluation

Common NLP/machine learning/AI approach



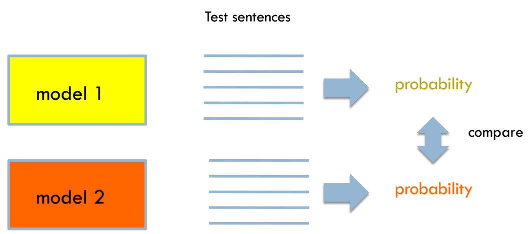
Evaluation



Ideas?

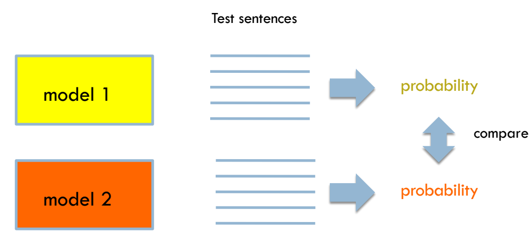
Evaluation

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



Evaluation

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.

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

$$prob(w_{1..n}) = \prod_{i=1}^{n} p(w_i | w_{1..i-1})$$

average the probabilities

geometric mean

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

Calculating perplexity in practice

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

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

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

What is this?

$$= -\frac{\sum_{i=1}^{n} \log p(w_i|w_{1..i-1})}{n}$$

Calculating perplexity in practice

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

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

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

Average logprob per word!

$$= -\frac{\sum_{i=1}^{n} \log p(w_i|w_{1..i-1})}{n}$$

Calculating perplexity

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

$$= 10^{-\frac{\sum_{i=1}^{n} \log_{10} p(w_i|w_{1..i-1})}{n}}$$

- 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 think today is a good day to be me) =
P(I | <start> <start>) x
P(think | <start> I) x
P(today | I think) x
P(is | think today) x
P(a | today is) x
P(good | is a) x
…

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 …
… x y d …
but never: xyz

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 … (100 times)
… x y d … (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 | xy) = 1/3? *reduce*
p(d | xy) = 2/3? *reduce*
p(z | xy) = 0/3? *increase*

$$p(c \mid a\ b) = \frac{\text{count}(a\ b\ c)}{\text{count}(a\ b)}$$

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

xya	1	1/3	2	2/29
xyb	0	0/3	1	1/29
xyc	0	0/3	1	1/29
xyd	2	2/3	3	3/29
xye	0	0/3	1	1/29
…				
xyz	0	0/3	1	1/29
Total xy	3	3/3	29	29/29

Add-one (Laplacian) smoothing

300 observations instead of 3 – better data, less smoothing

xya	100	100/300	101	101/326
xyb	0	0/300	1	1/326
xyc	0	0/300	1	1/326
xyd	200	200/300	201	201/326
xye	0	0/300	1	1/326
…				
xyz	0	0/300	1	1/326
Total xy	300	300/300	326	326/326

Add-one (Laplacian) smoothing

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

xya	1	1/3	2	2/29
xyb	0	0/3	1	1/29
xyc	0	0/3	1	1/29
xyd	2	2/3	3	3/29
xye	0	0/3	1	1/29
…				
xyz	0	0/3	1	1/29
Total xy	3	3/3	29	29/29

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

- add one smoothing thinks it is very likely to see a novel event

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

			modification	probability
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 λ = 0.01?

- This gives much less probability to novel events

see the abacus	1	1/3	1.01	1.01/203
see the abbot	0	0/3	0.01	0.01/203
see the abduct	0	0/3	0.01	0.01/203
see the above	2	2/3	2.01	2.01/203
see the Abram	0	0/3	0.01	0.01/203
…			0.01	0.01/203
see the zygote	0	0/3	0.01	0.01/203
Total	3	3/3	203	