N-gram Language Models

COMP90042

Natural Language Processing

Lecture 3

Semester 1 2022 Week 2

Jey Han Lau

THE UNIVERSITY OF MELBOURNE

Language Models

- One NLP application is about *explaining language*
  - Why some sentences are more **fluent** than others
- E.g. in speech recognition:
  - *recognise speech* > *wreck a nice beach*
- We measure 'goodness' using **probabilities** estimated by language models
- Language model can also be used for **generation**

↳ N-gram

InferKit DEMO

7497 / 10000
weekly free characters

Sign In

Generate Options

Learn more in the docs.

Length to generate

500

Start at beginning

Advanced Settings »

Donald Trump decides to become a vegan.

That's according to the diet of the President of the United States, who revealed his new eating habits Monday at a Mississippi campaign rally for Sen. Cindy Hyde-Smith, whom he endorsed over Democrat Mike Espy.

Trump, 71, spoke to the crowd at the Tupelo Regional Airport about his years of eating "perfectly healthy," including steaks and other meats.

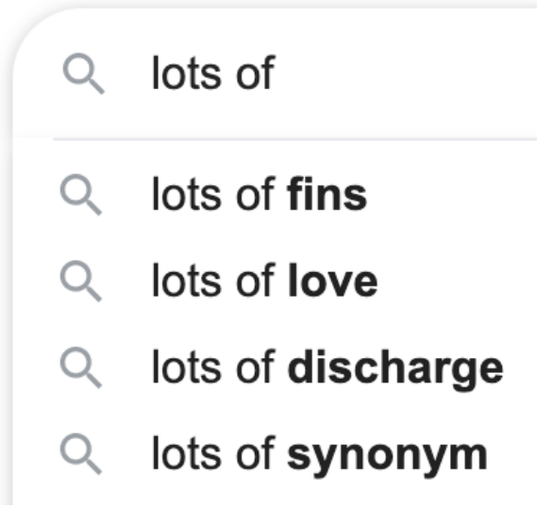
But now he's turned over a new leaf. "I'm not a fan of meat."

Trump told the crowd he now eats "mostly fish, like quite a bit of fish," and

Generate Text

Language Models

- Useful for
  - Query completion
  - Optical character recog.



  - Other generation tasks
    - Machine translation
    - Summarisation
    - Dialogue systems
- Nowadays pretrained language models are the backbone of modern NLP systems

Outline

- Deriving *n*-gram language models
- Smoothing to deal with sparsity

Probabilities: Joint to Conditional

Our goal is to get a probability for an arbitrary sequence of *m* words

$$P(w_1, w_2, \ldots, w_m)$$

First step is to apply the chain rule to convert joint probabilities to conditional ones

$$P(w_1, w_2, \ldots, w_m) = P(w_1)P(w_2 \mid w_1)P(w_3 \mid w_1, w_2) \ldots$$
$$P(w_m \mid w_1, \ldots, w_{m-1})$$

The Markov Assumption

(stat est.)
Too hard to find the prob. of such 19 words in that order in corpus.
i.e. 20 words in a sentence. $p(w_{20}|w_1 \dots w_{19})$

Still intractable, so make a simplifying assumption:

$$P(w_i \mid w_1 \dots w_{i-1}) \approx P(w_i \mid w_{i-n+1} \dots w_{i-1})$$

just look the fixed prior context (i.e. 3)

For some small *n*

When *n* = 1, a unigram model (bag-of-words model)

↓ compute each word independently

throw the sentence meaning away / just care some specific word

$$P(w_1, w_2, .. w_m) = \prod_{i=1}^{m} P(w_i)$$

w_i
the dog barks

When *n* = 2, a bigram model

$$P(w_1, w_2, .. w_m) = \prod_{i=1}^{m} P(w_i \mid w_{i-1})$$

w_i
the dog barks

When *n* = 3, a trigram model

$$P(w_1, w_2, .. w_m) = \prod_{i=1}^{m} P(w_i \mid w_{i-2} w_{i-1})$$

w_i
the dog barks

Maximum Likelihood Estimation

How do we calculate the probabilities? Estimate based on counts in our corpus:

For unigram models,

How many times you see "barks" in the corpus

$$P(w_i) = \frac{C(w_i)}{M}$$

$\frac{C(\text{barks})}{M}$

$M \rightarrow$ All word tokens in corpus

For bigram models,

How many times "dog barks" in yr corpus

$$P(w_i \mid w_{i-1}) = \frac{C(w_{i-1}w_i)}{C(w_{i-1})}$$

$\frac{C(\text{dog barks})}{C(\text{dog})}$

↳ how many times "dog" showed up (pre-condition)

For n-gram models generally,

$$P(w_i^{20} \mid w_{i-n+1}^{18} \ldots w_{i-1}^{19}) = \frac{C(w_{i-n+1} \ldots w_i)_{18.19.20}}{C(w_{i-n+1} \ldots w_{i-1})_{18.19}}$$

if $n=3$, total word in sentence is 20

Book-ending Sequences

- Special tags used to denote start and end of sequence
  - <s> = sentence start
  - </s> = sentence end

Trigram example

Corpus:

yes no no no no yes
no no no yes yes yes no

What is the probability of

<s> <s> *yes no no yes* </s>

under a trigram language model?

P(yes no no yes) =

P(*yes* | <*s*> <*s*>) ×

P(*no* | <*s*> *yes*) ×

P(*no* | *yes no*) ×

P(*yes* | *no no*) ×

P(</*s*> | *no yes*)

Need to predict </s> because it's the end of sentence!

Corpus:

<s> <s> *yes no no no no yes* </s>
<s> <s> *no no no yes yes yes no* </s>

Compute: P(*yes no no yes*)

$$P(w_i \mid w_{i-2}, w_{i-1}) = \frac{C(w_{i-2}, w_{i-1}, w_i)}{C(w_{i-2}, w_{i-1})}$$

Corpus:

① <s> <s> *yes no no no no yes* </s>
② <s> <s> *no no no yes yes yes no* </s>

Compute: P(*yes no no yes*)

$$P(w_i \mid w_{i-2}, w_{i-1}) = \frac{C(w_{i-2}, w_{i-1}, w_i)}{C(w_{i-2}, w_{i-1})}$$

P(*yes* | <s> <s>)	1/2	<s> <s> *yes* <s> <s> *no*

Corpus:

<s> <s> *yes no no no no yes* </s>
<s> <s> *no no no yes yes yes no* </s>

Compute: P(*yes no no yes*)

$$P(w_i \mid w_{i-2}, w_{i-1}) = \frac{C(w_{i-2}, w_{i-1}, w_i)}{C(w_{i-2}, w_{i-1})}$$

P(*yes* | <s> <s>)	1/2	<s> <s> *yes* <s> <s> *no*
P(*no* | <s> *yes*)	1/1	<s> *yes no*

Corpus:

<s> <s> *yes no no no no yes* </s>
<s> <s> *no no no yes yes yes no* </s>

Compute: P(*yes no no yes*)

$$P(w_i \mid w_{i-2}, w_{i-1}) = \frac{C(w_{i-2}, w_{i-1}, w_i)}{C(w_{i-2}, w_{i-1})}$$

P(*yes* | <s> <s>)	1/2	<s> <s> *yes* <s> <s> *no*
P(*no* | <s> *yes*)	1/1	<s> *yes no*
P(*no* | *yes no*)	1/2	*yes no no* *yes no* </s>

Corpus:

<s> <s> *yes no no no no yes* </s>
<s> <s> *no no no yes yes yes no* </s>

Compute: P(*yes no no yes*)

$$P(w_i \mid w_{i-2}, w_{i-1}) = \frac{C(w_{i-2}, w_{i-1}, w_i)}{C(w_{i-2}, w_{i-1})}$$

P(*yes* | <s> <s>)	1/2	<s> <s> *yes* <s> <s> *no*
P(*no* | <s> *yes*)	1/1	<s> *yes no*
P(*no* | *yes no*)	1/2	*yes no no* *yes no* </s>
P(*yes* | *no no*)	? 2/5	

PollEv.com/jeyhanlau569

Corpus:

<s> <s> *yes no no no no yes* </s>
<s> <s> *no no no yes yes yes no* </s>

Compute: P(*yes no no yes*)

$$P(w_i \mid w_{i-2}, w_{i-1}) = \frac{C(w_{i-2}, w_{i-1}, w_i)}{C(w_{i-2}, w_{i-1})}$$

P(*yes* | <s> <s>)	1/2	<s> <s> *yes* <s> <s> *no*
P(*no* | <s> *yes*)	1/1	<s> *yes no*
P(*no* | *yes no*)	1/2	*yes no no* *yes no* </s>
P(*yes* | *no no*)	2/5	no no no no no no no no yes no no no no no yes

Corpus:

<s> <s> *yes no no no no yes* </s>
<s> <s> *no no no yes yes yes no* </s>

Compute: P(*yes no no yes*)

$$P(w_i \mid w_{i-2}, w_{i-1}) = \frac{C(w_{i-2}, w_{i-1}, w_i)}{C(w_{i-2}, w_{i-1})}$$

P(*yes* | <s> <s>)	1/2	<s> <s> *yes* <s> <s> *no*
P(*no* | <s> *yes*)	1/1	<s> *yes no*
P(*no* | *yes no*)	1/2	*yes no no* *yes no* </s>
P(*yes* | *no no*)	2/5	no no no no no no no no yes no no no no no yes
P(</s> | *no yes*)	1/2	no yes </s> no yes yes

Corpus:

<s> <s> *yes no no no no yes* </s>
<s> <s> *no no no yes yes yes no* </s>

Compute: P(*yes no no yes*) = ½ × 1 × ½ × ⅖ × ½ = 0.05

P(*yes* | <s> <s>)	1/2	<s> <s> *yes* <s> <s> *no*
P(*no* | <s> *yes*)	1/1	<s> *yes no*
P(*no* | *yes no*)	1/2	*yes no no* *yes no* </s>
P(*yes* | *no no*)	2/5	no no no no no no no no yes no no no no no yes
P(</s> | *no yes*)	1/2	no yes </s> yes no </s>

Several Problems

- Language has long distance effects — need large *n*
  - *The lecture/s that took place last week was/were on preprocessing.*
- Resulting probabilities are often very small
  - Use log probability to avoid numerical underflow

summing up

- What about unseen *n*-grams?

whole term = 0 if $P(w_2|w_1) = 0$

$$P(w_1, w_2, \ldots, w_m) = P(w_1 \mid < s >) \times P(w_2 \mid w_1) \ldots$$

  - Need to smooth the LM!

Smoothing

Smoothing

- Basic idea: give events you've never seen before some probability
- Must be the case that *P*(everything) = 1
- Many different kinds of smoothing
  - Laplacian (add-one) smoothing
  - Add-*k* smoothing
  - Absolute discounting
  - Kneser-Ney
  - And others…

Laplacian (Add-one) Smoothing

- Simple idea: pretend we've seen each *n*-gram once more than we did.

added one pseudo count for each word you've seen in the corpus.

For unigram models (**V**= the vocabulary), size of

$$P_{add1}(w_i) = \frac{C(w_i) + 1}{M + |V|}$$

For bigram models,

$$P_{add1}(w_i \mid w_{i-1}) = \frac{C(w_{i-1}w_i) + 1}{C(w_{i-1}) + |V|}$$

Add-one Example

is only ever used as a prior context → never add pseudo count to <s>

<s> *the rat ate the cheese* </s>

↳ add one coz we have to estimate it sometimes

What's the bigram probability *P(ate* | *rat*) under add-one smoothing?

$$= \frac{C(rat\ ate) + 1}{C(rat) + |V|} = \frac{2}{6}$$

V = { the, rat, ate, cheese, </s> }

What's the bigram probability *P(ate* | *cheese*) under add-one smoothing?

$$= \frac{C(cheese\ ate) + 1}{C(cheese) + |V|} = \frac{1}{6}$$

<s> is not part of vocabulary because we never need to infer its conditional probability (e.g. P(<s> | …)) predict it

Recall: P(yes no no yes) =
P(*yes* | *<s> <s>*) × P(*no* | *<s> yes*) ×
P(*no* | *yes no*) × P(*yes* | *no no*) ×
P(*</s>* | *no yes*)

Add-*k* Smoothing

- Adding one is often too much
- Instead, add a fraction *k*
- AKA Lidstone Smoothing, or Add-α Smoothing

$$P_{addk}(w_i \mid w_{i-1}, w_{i-2}) = \frac{C(w_{i-2}, w_{i-1}, w_i) + k}{C(w_{i-2}, w_{i-1}) + k|V|}$$

- Have to choose *k*

Taking the effective counts from observed bigrams to unseen ones

maintain the prob. dist.

Lidstone Smoothing

real counts ↑

effective counts taken ↑

Prior Context = *alleged*

- 5 observed bi-grams
- 2 unobserved bi-grams

a bit smaller

	counts	unsmoothed probability	Lidstone smoothing, $\alpha = 0.1$ effective counts	smoothed probability
impropriety	8	0.4	7.826	0.391
offense	5	0.25	4.928	0.246
damage	4	0.2	3.961	0.198
deficiencies	2	0.1	2.029	0.101
outbreak	1	0.05	1.063	0.053
infirmity	0	0	0.097	0.005
alleged	0	0	0.097	0.005
	20	1.0	20	1.0

Alleged + [2nd-3rd]

pseudo counts given

a bit more

0.391 x 20

(0 + 0.1) / (20 + 7 x 0.1)

|V|

(8 + 0.1) / (20 + 7 x 0.1)

3 Absolute Discounting

take same effective counts

- 'Borrows' a **fixed** probability mass from observed n-gram counts
- Redistributes it to unseen n-grams

Absolute Discounting

Context = *alleged*

- 5 observed bi-grams
- 2 unobserved bi-grams

			Lidstone smoothing, $\alpha = 0.1$		Discounting, $d = 0.1$	
	counts	unsmoothed probability	effective counts	smoothed probability	effective counts	smoothed probability
impropriety	8	0.4	7.826	0.391	7.9	0.395
offense	5	0.25	4.928	0.246	4.9	0.245
damage	4	0.2	3.961	0.198	3.9	0.195
deficiencies	2	0.1	2.029	0.101	1.9	0.095
outbreak	1	0.05	1.063	0.053	0.9	0.045
infirmity	0	0	0.097	0.005	0.25	0.013
alleged	0	0	0.097	0.005	0.25	0.013
	20	1.0	20	1.0	20	1.0

factor

8 - 0.1

8 - 0.1
5 - 0.1
4 - 0.1
2 - 0.1
1 - 0.1

(0.1 x 5) / 2

two unseen ones

0.1 have been taken 5 times

0.25 / 20

effective counts / total tokens

④ Backoff

freq.↑ weight↑.

- Absolute discounting redistributes the probability mass **equally** for all unseen n-grams
  ↳ Not realistic
- Katz Backoff: redistributes the mass based on a **lower order** model (e.g. unigram) decide how much mass we give to the unseen bigrams

$$P_{katz}(w_i|w_{i-1}) = \begin{cases} \frac{C(w_{i-1},w_i)-D}{C(w_{i-1})}, & \text{if } C(w_{i-1},w_i) > 0 \\ \alpha(w_{i-1}) \times \frac{P(w_i)}{\sum_{w_j:C(w_{i-1},w_j)=0} P(w_j)}, & \text{otherwise} \end{cases}$$

for seen bigrams

what you have taken from the seen bigrams

unigram probability for w_i
e.g. P(infirmity)

weight each unseen bigram based on unigram

sum unigram probabilities for all words that do not co-occur with context w_{i-1}
unseen e.g. P(infirmity) + P(alleged)

the amount of probability mass that has been discounted for context w_{i-1}
((0.1 x 5) / 20 in previous slide)

Issues with Katz Backoff

$$P_{katz}(w_i|w_{i-1}) = \begin{cases} \frac{C(w_{i-1},w_i)-D}{C(w_{i-1})}, & \text{if } C(w_{i-1}, w_i) > 0 \\ \alpha(w_{i-1}) \times \frac{P(w_i)}{\sum_{w_j:C(w_{i-1},w_j)=0} P(w_j)}, & \text{otherwise} \end{cases}$$

- *I can't see without my reading* ___ (What's most likely next word?)
- C(*reading*, *glasses*) = C(*reading*, *Francisco*) = 0 (bigram)
- C(*Fransciso*) > C(*glasses*) (Unigram; in a corpus (assumed))
- Katz backoff will give higher probability to *Francisco* (But actually doesn't make sense)

capture ability better when redistributing

⑤ Kneser-Ney Smoothing

- Redistribute probability mass based on the **versatility** of the lower order n-gram
- AKA "continuation probability"
- What is versatility? how often a word co-occur with other unique word.

  - High versatility -> co-occurs with a lot of unique words, e.g. glasses
    - men's glasses, black glasses, buy glasses, etc
  - Low versatility -> co-occurs with few unique words, e.g. francisco
    - san francisco

Kneser-Ney Smoothing

seen bigram

$$P_{KN}(w_i \mid w_{i-1}) = \begin{cases} \dfrac{C(w_{i-1}, w_i) - D}{C(w_{i-1})}, & \text{if } C(w_{i-1}, w_i) > 0 \\ \beta(w_{i-1}) P_{cont}(w_i), & \text{otherwise} \end{cases}$$

α =

the amount of probability mass that has been discounted for context w_{i-1}

有多少种 word 不同种类

how many unique prior context word for w_i

$$P_{cont}(w_i) = \frac{|\{w_{i-1} : C(w_{i-1}, w_i) > 0\}|}{\sum_{w_j} |\{w_{j-1} : C(w_{j-1}, w_j) > 0\}|}$$

看 w_i 前面 w_i 有多少种不同的 w_{i-1}，加起来 ← All the diff w_i, count the freq. of versatility

C(word type of w_{i-1})

- Intuitively the numerator of P_{cont} counts the number of unique w_{i-1} that co-occurs with w_i
- High continuation counts for glasses
- Low continuation counts for Franciso

6 Interpolation

- A better way to combine different orders of *n*-gram models

sum = unigram + bigram + trigram

- Interpolated trigram model:

$$P_{IN}(w_i \mid w_{i-1}, wi-2) = \lambda_3 P_3(w_i \mid w_{i-2}, w_{i-1}) + \lambda_2 P_2(w_i \mid w_{i-1}) + \lambda_1 P_1(w_i)$$

Learned based on held out data

$$\lambda_1 + \lambda_2 + \lambda_3 = 1$$

A Final Word

- *N*-gram language models are a simple but effective way to capture the predictability of language
- Can be trained in an unsupervised fashion, scalable to large corpora
- Require smoothing to be effective
- Modern language models uses neural networks (lecture 8)

Reading

- E18 Chapter 6 (skip 6.3)

Events Confirmed	Week	Date, Time, Location
Summerfest Welcome Back Expo	1	1-Mar-22, 2-5pm 2-Mar-22, 2-5pm 3-Mar-22, 2-5pm 4-Mar-22, 2-5pm
Welcome event with DSSS & HackMelbourne	2	7-Mar-22, 11:30am
Google Career Discussion	2	8-Mar-22, 5 - 6:30pm
Spaghetti Bridge Building Comp Collab	2	11-Mar-22, 3-5:30pm
Robogals collab industry week	3	this whole week
Google Women in Tech clubs session	3	16-Mar-22, 5-5:45pm
Macquarie Grad Program Event	3	TBD

Events planned (wk4 onwards):

- Annual Hackathon
- Workshops for CV and job seeking
- High Tea networking
- Industry Panels
- Free lunch on campus
- Study sessions
- Tech-related clubs collaboration
- SWOT cake
- ...

Other opportunities:

- Weekly job posts
- Tech-related events/competitions
- Volunteering opportunities
- ...

WOMEN IN TECH

The University of Melbourne

JOIN US HERE!