

N-gram Language Models

COMP90042

Natural Language Processing

Lecture 3

Semester 1 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**



Write a fictional news story where Donald Trump became a vegan.



In a surprising turn of events, former US President Donald Trump has announced that he has adopted a vegan lifestyle. The announcement came during a press conference at his Mar-a-Lago estate in Florida.



"I've always been a meat and potatoes kind of guy, but after some reflection, I've decided to make a change," said Trump. "I've been reading a lot about the benefits of a plant-based diet, and I think it's time for me to give it a try."

The announcement has been met with mixed reactions, with some of Trump's supporters expressing disappointment and others applauding his decision. Many have also expressed skepticism, with some suggesting that the move is a publicity stunt.

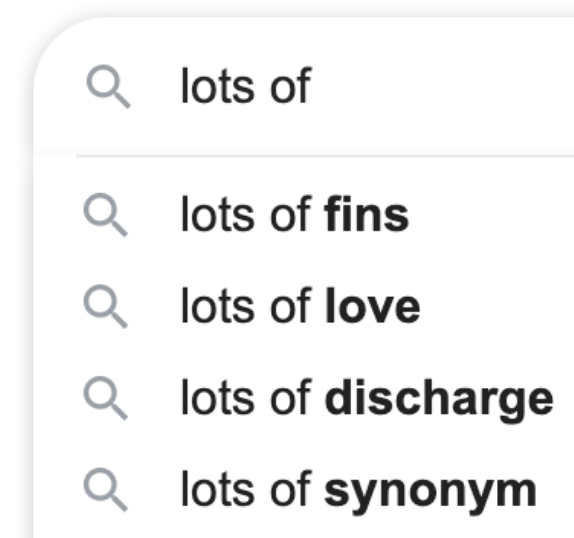
Trump, however, has been adamant that he is committed to the vegan lifestyle, and has even hired a personal chef to help him with his new diet. He has also been vocal about the health benefits of a vegan diet, and has encouraged others to consider making the switch.

"I know this is a big change for me, but I'm excited to see the benefits it will bring," said Trump. "I hope that others will join me in making healthier choices and taking care of our bodies."

Only time will tell if Trump will stick with his new vegan lifestyle, but one thing is for certain: this unexpected move has once again put him in the headlines.

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

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

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

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

The Markov Assumption

Still intractable, so make a simplifying assumption:

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

For some small n

When $n = 1$, a unigram model

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

the dog ^{w_i} barks

When $n = 2$, a bigram model

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

the ^{w_i} dog barks

When $n = 3$, a trigram model

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

the ^{w_i} dog barks

Maximum Likelihood Estimation

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

For unigram models,

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

← Total number of word tokens in corpus

For bigram models,

$$P(w_i | w_{i-1}) = \frac{C(w_{i-1}w_i)}{C(w_{i-1})} \quad \frac{C(\text{dog barks})}{C(\text{dog})}$$

For n-gram models generally,

$$P(w_i | w_{i-n+1} \dots w_{i-1}) = \frac{C(w_{i-n+1} \dots w_i)}{C(w_{i-n+1} \dots w_{i-1})}$$

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

yes no no yes

under a trigram language model?

$P(\text{yes no no yes}) =$

$P(\text{yes} \mid \langle s \rangle \langle s \rangle) \times$

$P(\text{no} \mid \langle s \rangle \text{yes}) \times$

$P(\text{no} \mid \text{yes no}) \times$

$P(\text{yes} \mid \text{no no}) \times$

$P(\langle /s \rangle \mid \text{no yes})$

 Need to predict $\langle /s \rangle$ 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: $P(\text{yes no no yes})$

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

Corpus:

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

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

Compute: $P(\text{yes no no yes})$

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

$P(\text{yes} | \text{<s> <s>})$

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: $P(\text{yes no no yes})$

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

$P(\text{yes} \text{<s> <s>})$	1/2	<s> <s> yes <s> <s> no
$P(\text{no} \text{<s> yes})$	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: $P(\text{yes no no yes})$

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

$P(\text{yes} \text{<s> <s>})$	1/2	<s> <s> <i>yes</i> <s> <s> <i>no</i>
$P(\text{no} \text{<s> yes})$	1/1	<s> <i>yes no</i>
$P(\text{no} \text{yes no})$	1/2	<i>yes no no</i> <i>yes no </s></i>

Corpus:

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

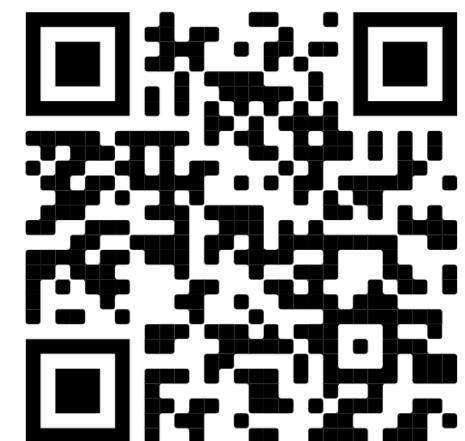
<s> <s> no no no yes yes yes no </s>

Compute: $P(\text{yes no no yes})$

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

$P(\text{yes} \text{<s> <s>})$	1/2	<i><s> <s> yes <s> <s> no</i>
$P(\text{no} \text{<s> yes})$	1/1	<i><s> yes no</i>
$P(\text{no} \text{yes no})$	1/2	<i>yes no no yes no </s></i>
$P(\text{yes} \text{no no})$?	

[PollEv.com/jeyhanlau569](https://poll-ev.com/jeyhanlau569)



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
- What about unseen n -grams?
 - $P(w_1, w_2, \dots, w_m) = P(w_1 \mid \langle s \rangle) \times P(w_2 \mid w_1) \dots$
 - whole term = 0 if $P(w_2 \mid w_1) = 0$
 - Need to smooth the LM!

Smoothing

Smoothing

- Basic idea: give events you've never seen before some probability
- Must be the case that $P(\text{everything}) = 1$
- 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.

For unigram models (\mathbf{V} = the vocabulary),

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

For bigram models,

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

Add-one Example

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

What's the bigram probability $P(\text{ate} \mid \text{rat})$ under add-one smoothing?

$$= \frac{C(\text{rat ate}) + 1}{C(\text{rat}) + |V|} = \frac{2}{6} \quad V = \{ \text{the, rat, ate, cheese, } \langle /s \rangle \}$$

What's the bigram probability $P(\text{ate} \mid \text{cheese})$ under add-one smoothing?

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

<s> is not part of vocabulary because we never need to infer its conditional probability (e.g. $P(\langle s \rangle \mid \dots)$)

Recall: $P(\text{yes no no yes}) =$

$P(\text{yes} \mid \langle s \rangle \langle s \rangle) \times P(\text{no} \mid \langle s \rangle \text{yes}) \times$

$P(\text{no} \mid \text{yes no}) \times P(\text{yes} \mid \text{no no}) \times$

$P(\langle /s \rangle \mid \text{no yes})$

Add- k Smoothing

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

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

- Have to choose k

Lidstone Smoothing

Context = *alleged*

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

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

$$0.391 \times 20$$

$$(0 + 0.1) / (20 + 7 \times 0.1)$$

$$(8 + 0.1) / (20 + 7 \times 0.1)$$

Absolute Discounting

- ‘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
<i>impropriety</i>	8	0.4	7.826	0.391	7.9	0.395
<i>offense</i>	5	0.25	4.928	0.246	4.9	0.245
<i>damage</i>	4	0.2	3.961	0.198	3.9	0.195
<i>deficiencies</i>	2	0.1	2.029	0.101	1.9	0.095
<i>outbreak</i>	1	0.05	1.063	0.053	0.9	0.045
<i>infirmity</i>	0	0	0.097	0.005	0.25	0.013
<i>alleged</i>	0	0	0.097	0.005	0.25	0.013
	20	1.0	20	1.0	20	1.0

$8 - 0.1$
 $(0.1 \times 5) / 2$
 $0.25 / 20$

Backoff

- Absolute discounting redistributes the probability mass **equally** for all unseen n-grams
- Katz Backoff: redistributes the mass based on a **lower order** model (e.g. unigram)

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

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

unigram probability for w_i
 e.g. $P(\text{infirmity})$

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 ____*
- $C(\text{reading}, \text{glasses}) = C(\text{reading}, \text{Francisco}) = 0$
- $C(\text{Francisco}) > C(\text{glasses})$
- Katz backoff will give higher probability to *Francisco*

Kneser-Ney Smoothing

- Redistribute probability mass based on the **versatility** of the lower order n-gram
- AKA “continuation probability”
- What is versatility?
 - 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

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

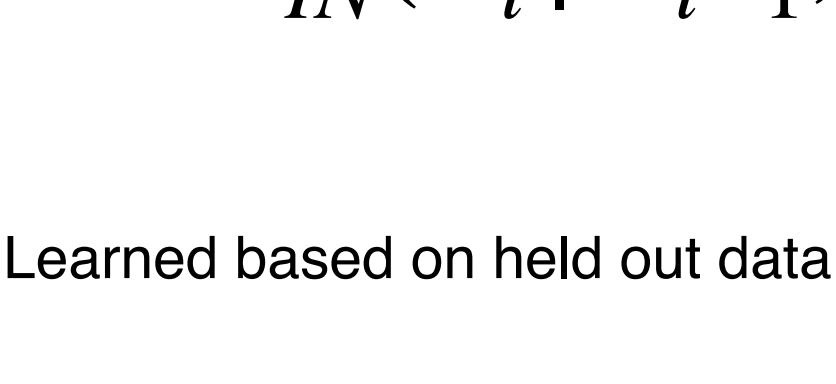
the amount of probability mass that has been discounted for context w_{i-1}

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

- Intuitively the numerator counts #unique preceding words of w_i
- Denominator sums all continuation counts for unseen bigrams
- High continuation counts for glasses
- Low continuation counts for Franciso

Interpolation

- A better way to combine different orders of n -gram models
- Interpolated trigram model:

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


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)