Language Modeling

2/2565: FRA501 Introduction to Natural Language Processing with Deep learning

Week 04

Paisit Khanarsa, Ph.D.

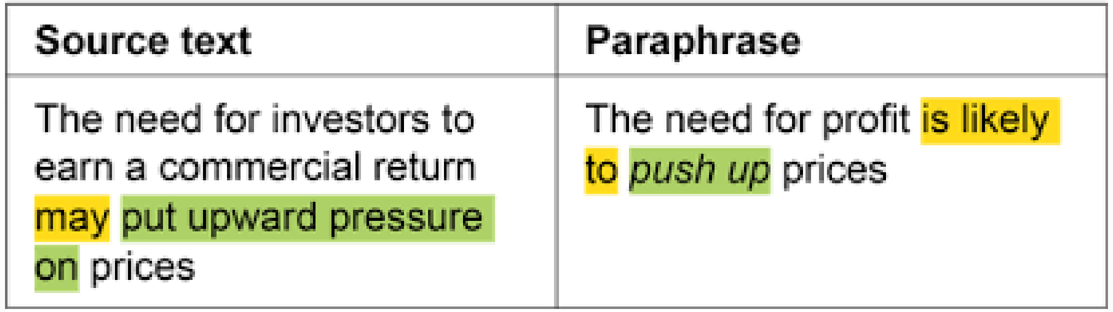
Institute of Field Robotics (FIBO), King Mongkut's University of Technology Thonburi

Outlines

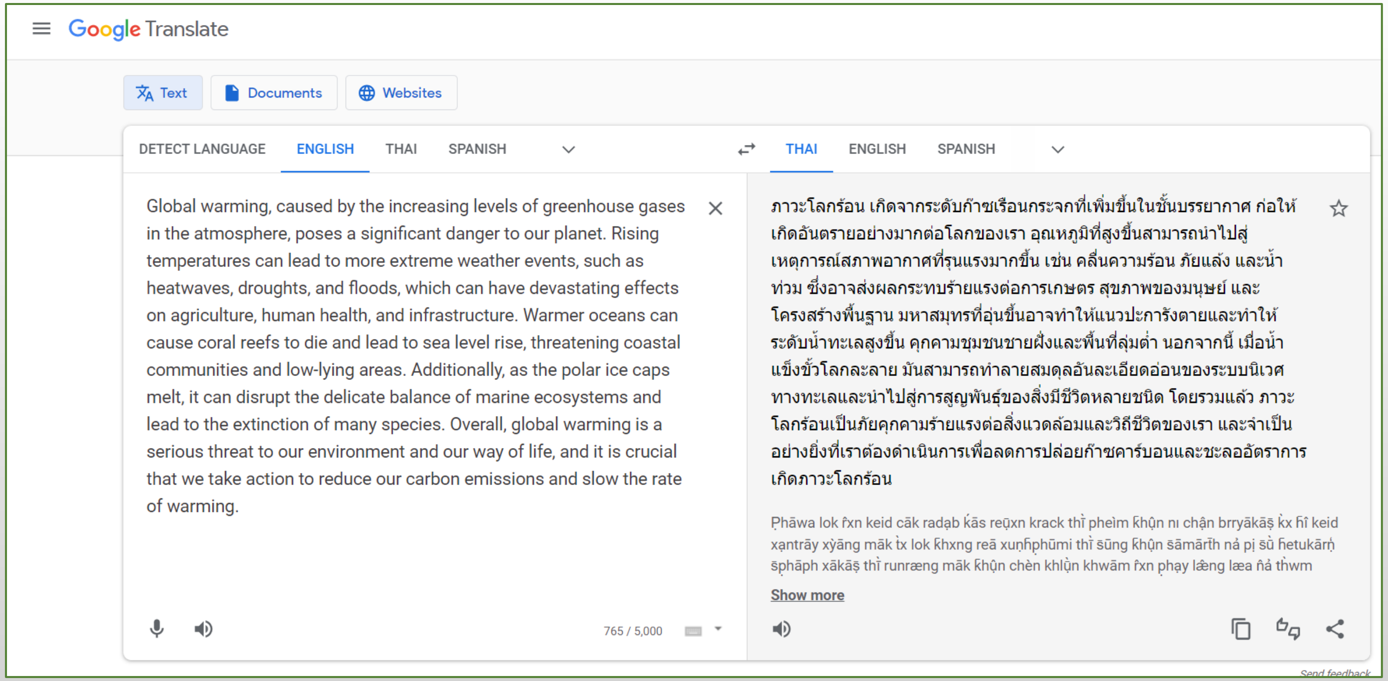
- Introduction
- N-grams
- Evaluation and Perplexity
- Smoothing
- Neural Language Model
- Demos

Introduction

- Language model’s goal is
  - To assign probability to a sentence
  - To predict the next word
- Example: Which sentence is more likely to occur?
  - “Do you live in Bangkok?”
  - “Live in Bangkok do you?”



Source text	Paraphrase
The need for investors to earn a commercial return may put upward pressure on prices	The need for profit is likely to *push up* prices



Global warming, caused by the increasing levels of greenhouse gases in the atmosphere, poses a significant danger to our planet. Rising temperatures can lead to more extreme weather events, such as heatwaves, droughts, and floods, which can have devastating effects on agriculture, human health, and infrastructure. Warmer oceans can cause coral reefs to die and lead to sea level rise, threatening coastal communities and low-lying areas. Additionally, as the polar ice caps melt, it can disrupt the delicate balance of marine ecosystems and lead to the extinction of many species. Overall, global warming is a serious threat to our environment and our way of life, and it is crucial that we take action to reduce our carbon emissions and slow the rate of warming.

ภาวะโลกร้อน เกิดจากระดับก๊าซเรือนกระจกที่เพิ่มขึ้นในชั้นบรรยากาศ ก่อให้เกิดอันตรายอย่างมากต่อโลกของเรา อุณหภูมิที่สูงขึ้นสามารถนำไปสู่เหตุการณ์สภาพอากาศที่รุนแรงมากขึ้น เช่น คลื่นความร้อน ภัยแล้ง และน้ำท่วม ซึ่งอาจส่งผลกระทบร้ายแรงต่อการเกษตร สุขภาพของมนุษย์ และโครงสร้างพื้นฐาน มหาสมุทรที่อุ่นขึ้นอาจทำให้แนวปะการังตายและทำให้ระดับน้ำทะเลสูงขึ้น คุกคามชุมชนชายฝั่งและพื้นที่ลุ่มต่ำ นอกจากนี้ เมื่อน้ำแข็งขั้วโลกละลาย มันสามารถทำลายสมดุลอันละเอียดอ่อนของระบบนิเวศทางทะเลและนำไปสู่การสูญพันธุ์ของสิ่งมีชีวิตหลายชนิด โดยรวมแล้ว ภาวะโลกร้อนเป็นภัยคุกคามร้ายแรงต่อสิ่งแวดล้อมและวิถีชีวิตของเรา และจำเป็นอย่างยิ่งที่เราต้องดำเนินการเพื่อลดการปล่อยก๊าซคาร์บอนและชะลออัตราการเกิดภาวะโลกร้อน

P̣hāwa lok r̂xn keid cāk radạb k̆ās reụ̄xn krack thī̀ pheìm k̄hụ̂n nı chận brryākāṣ̄ k̀x h̄ı̂ keid xạntrāy xỳāng māk t̀x lok k̄hxng reā xuṇh̄p̣hūmi thī̀ s̄ūng k̄hụ̂n s̄āmārt̄h nả pị s̄ū̀ h̄etukārṇ̒ s̄p̣hāph xākāṣ̄ thī̀ runræng māk k̄hụ̂n chèn khlụ̄̀n khwām r̂xn p̣hạy læ̂ng læa n̂ả th̀wm

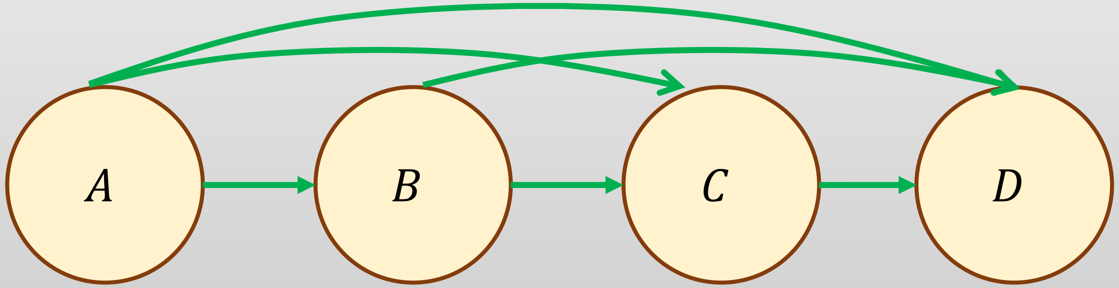
Introduction (cont.)

- How to compute this sentence probability?
  - S = “It was raining cat and dog yesterday”
  - What is $P(S)$

Introduction (cont.)

- How to compute this sentence probability?
  - S = “It was raining cat and dog yesterday”
  - What is $P(S)$

- Conditional probability
  - $P(B|A) = \frac{P(A,B)}{P(B)}$
  - $P(A,B,C,D) = P(A)P(B|A)P(C|A,B)P(D|A,B,C)$

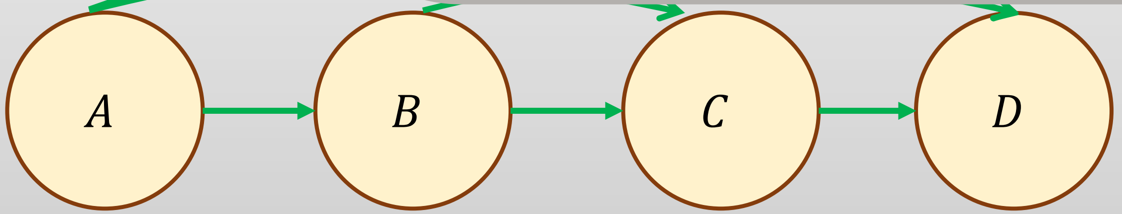


Introduction (cont.)

- How to cor
    - S = “It wa
    - What is P

- Conditional
    - $P(B|A) =$
    - $P(A, B, C,$

Example:
What is the probability of
$P(\text{It, was, raining, cats, and, dogs, yesterday})$?



Introduction (cont.)

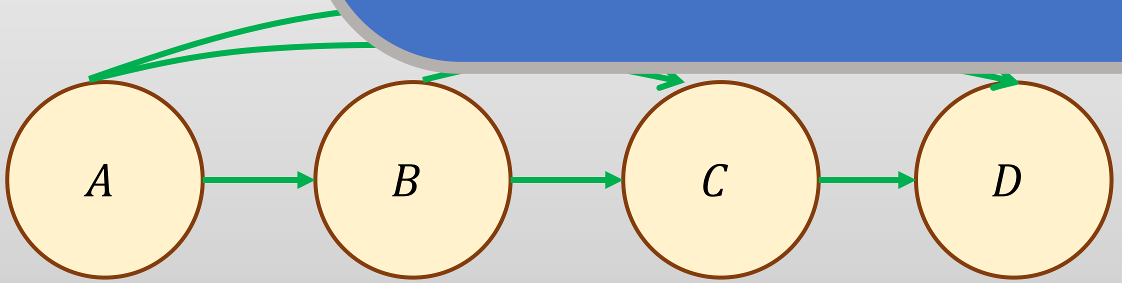
- How to cor
  - S = “It wa
  - What is P
- Conditional
  - $P(B|A) =$
  - $P(A, B, C,$

Example:

What is the probability of

$P(\text{It, was, raining, cats, and, dogs, yesterday})$?

$= P(\text{it}) \times P(\text{was}|\text{it}) \times P(\text{raining}|\text{it was}) \times$
$P(\text{cats}|\text{it was raning}) \times P(\text{and}|\text{It was raining cats}) \times$
$P(\text{dogs}|\text{it was raining cats and}) \times$
$P(\text{yerterday}|\text{it was tainig cats and dogs})$



Problem with full estimation

- Language is creative.
- New sentence are created all the time.
- Can not count all them

Problem with full estimation (cont.)

- Language is creative.
- New sentence are created all the time.
- Can not count all them

- Example
  - Training :
    - <s> I am a student . </s>
    - <s> I live in Bangkok . </s>
    - <s> I like to read . </s>
  - Test :
    - <s> I am a teacher . </s>
  - What is the probability of $P($<s> I am a teacher . </s>$)$?
    -

Problem with full estimation (cont.)

- Language is creative.
- New sentence are created all the time.
- Can not count all them

- Example
  - Training :
    - <s> I am a student . </s>
    - <s> I live in Bangkok . </s>
    - <s> I like to read . </s>
  - Test :
    - <s> I am a teacher . </s>
  - What is the probability of $P($<s> I am a teacher . </s>$)$?
    - $P($<s>$) \times P(\mathrm{i}|$<s>$) \times P(\mathrm{am}|$<s> I$) \times P(a|$<s> I am$) \times P($teacher|<s> I am a$) \times$ $P($.|<s> I am a teacher$) \times P($</s>|<s> I am a teacher .$)$

Problem with full estimation (cont.)

- Language is creative.
- New sentence are created all the time.
- Can not count all them

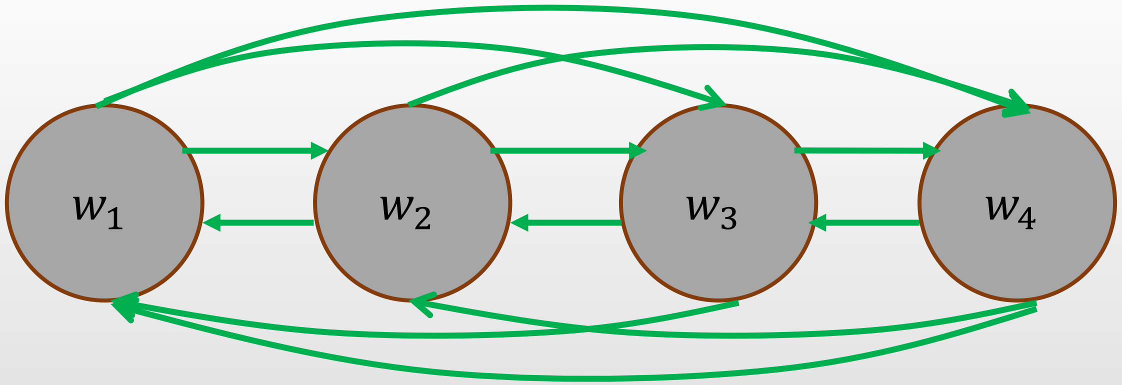
- Example
  - Training :
    - <s> I am a student . </s>
    - <s> I live in Bangkok . </s>
    - <s> I like to read . </s>
  - Test :
    - <s> I am a teacher . </s>
  - What is the probability of $P(\text{<s> I am a teacher . </s>})$?

Probability = 0

  - $P(\text{<s>}) \times P(\mathrm{i}|\text{<s>}) \times P(\mathrm{am}|\text{<s> I}) \times P(a|\text{<s> I am}) \times \boxed{P(\text{teacher}|\text{<s> I am a})} \times P(.|\text{<s> I am a teacher}) \times P(\text{</s>}|\text{<s> I am a teacher .})$

N-grams: A probability of the next word

- Markov Assumption
  - Predict the probability of the next word without looking too far into the past
  - Relation of #words: unigram, bigrams, trigrams or n-grams



N-grams: A probability of the next word (cont.)

- Markov Assumption
  - Predict the probability of the next word without looking too far into the past
  - Relation of #words: unigram, bigrams, trigrams or n-grams
  - Example: bigram
    - $P(F|A,B,C,D,E) \sim P(F|E)$
    - $S =$ There are ten students in the class
    - $P(\text{class}|\text{There are ten students in the})$
      - Unigrams $\sim P(\text{class})$
      - Bigrams $\sim P(\text{class}|\text{the})$
      - Trigrams $\sim P(\text{class}|\text{in the })$

N-grams: A probability of the next word (cont.)

- Full estimation

$$P(\text{It}, \text{was}, \text{raining}, \text{cats}, \text{and}, \text{dogs}, \text{yesterday}) = P(\text{it}) \times P(\text{was}|\text{it}) \times P(\text{raining}|\text{it was}) \times P(\text{cats}|\text{it was raning}) \times P(\text{and}|\text{It was raining cats}) \times P(\text{dog}|\text{it was raining cats and}) \times P(\text{yerterday}|\text{it was tainig cats and dogs})$$

- Trigrams(Markov assumption)

$$P(\text{It}, \text{was}, \text{raining}, \text{cats}, \text{and}, \text{dogs}, \text{yesterday}) =$$

N-grams: A probability of the next word (cont.)

- Full estimation

$$P(\text{It}, \text{was}, \text{raining}, \text{cats}, \text{and}, \text{dogs}, \text{yesterday}) = P(\text{it}) \times P(\text{was}|\text{it}) \times P(\text{raining}|\text{it was}) \times$$
$$P(\text{cats}|\text{it was raning}) \times P(\text{and}|\text{It was raining cats}) \;\times P(\text{dog}|\text{it was raining cats and}) \times$$
$$P(\text{yerterday}|\text{it was tainig cats and dogs})$$

- Trigrams(Markov assumption)

$$P(\text{It}, \text{was}, \text{raining}, \text{cats}, \text{and}, \text{dogs}, \text{yesterday}) = P(\text{it}) \times P(\text{was}|\text{it}) \times P(\text{raining}|\text{it was}) \times$$
$$P(\text{cats}|\text{was raning}) \times P(\text{and}|\text{raining cats}) \;\times P(\text{dog}s|\text{cats and}) \times P(\text{yerterday}|\text{and dog}s)$$

N-grams: A probability of the next word (cont.)

- Full estimation

$$P(\text{It, was, raining, cats, and, dogs, yesterday}) = P(\text{it}) \times P(\text{was}|\text{it}) \times P(\text{raining}|\text{it was}) \times P(\text{cats}|\text{it was raning}) \times P(\text{and}|\text{It was raining cats}) \times P(\text{dog}|\text{it was raining cats and}) \times P(\text{yerterday}|\text{it was tainig cats and dogs})$$

- Trigrams(Markov assumption)

Add start (<s>) & stop(<\s>)

$$P(\text{<s>, It, was, raining, cats, and, dogs, yesterday, </s>}) = P(\text{<s>}) \times P(\text{it}|\text{<s>}) \times P(\text{was}|\text{<s> it}) \times P(\text{raining}|\text{it was}) \times P(\text{cats}|\text{was raning}) \times P(\text{and}|\text{raining cats}) \times P(\text{dogs}|\text{cats and}) \times P(\text{yerterday}|\text{and dogs}) \times P(\text{</s>}|\text{dog yesterday})$$

N-grams: Example

- Estimate Bigrams Probability
    - <s> I am Sam </s>
    - <s> Sam I am </s>
    - <s> I am not Sam </s>

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

Bigrams Unit	Bigrams Probability

N-grams: Example (cont.)

- Estimate Bigrams Probability
  - <s> I am Sam </s>
  - <s> Sam I am </s>
  - <s> I am not Sam </s>

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

Bigrams Unit	Bigrams Probability
$P(\text{i}|\text{<s>})$	
$P(\text{am}|\text{i})$	
$P(\text{sam}|\text{am})$	
$P(\text{</s>}|\text{sam})$	
$P(\text{sam}|\text{<s>})$	
$P(\text{i}|\text{sam})$	
$P(\text{</s>}|\text{am})$	
$P(\text{not}|\text{am})$	
$P(\text{sam}|\text{not})$	

N-grams: Example (cont.)

- Estimate Bigrams Probability
  - <s> I am Sam </s>
  - <s> Sam I am </s>
  - <s> I am not Sam </s>

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

Bigrams Unit	Bigrams Probability
$P(\text{i}\|\text{<s>})$	2/3 = 0.67
$P(\text{am}\|\text{i})$	3/3 = 1
$P(\text{sam}\|\text{am})$	1/3 = 0.33
$P(\text{</s>}\|\text{sam})$	2/3 = 0.67
$P(\text{sam}\|\text{<s>})$	1/3 = 0.33
$P(\text{i}\|\text{sam})$	1/3 = 0.33
$P(\text{</s>}\|\text{am})$	1/3 = 0.33
$P(\text{not}\|\text{am})$	1/3 = 0.33
$P(\text{sam}\|\text{not})$	1/1 = 1

N-grams: Example (cont.

- Estimate Bigrams Probability
  - <s> I am Sam </s>
  - <s> Sam I am </s>
  - <s> I am not Sam </s>

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

Bigrams Unit	Bigrams Probability
$P(\text{i}|\text{<s>})$	2/3 = 0.67
$P(\text{am}|\text{i})$	3/3 = 1
$P(\text{sam}|\text{am})$	1/3 = 0.33
$P(\text{</s>}|\text{sam})$	2/3 = 0.67
$P(\text{sam}|\text{<s>})$	1/3 = 0.33
$P(\text{i}|\text{sam})$	1/3 = 0.33
$P(\text{</s>}|\text{am})$	1/3 = 0.33
$P(\text{not}|\text{am})$	1/3 = 0.33
$P(\text{sam}|\text{not})$	1/1 = 1

Sentence	Bigrams Probability
P(<s> I am Sam </s>)	
P(<s> Sam I am </s>)	
P(<s> I am not Sam </s>)	

N-grams: Example (cont.

- Estimate Bigrams Probability
  - <s> I am Sam </s>
  - <s> Sam I am </s>
  - <s> I am not Sam </s>

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

Bigrams Unit	Bigrams Probability
$P(\text{i} \mid \text{<s>})$	2/3 = 0.67
$P(\text{am} \mid \text{i})$	3/3 = 1
$P(\text{sam} \mid \text{am})$	1/3 = 0.33
$P(\text{</s>} \mid \text{sam})$	2/3 = 0.67
$P(\text{sam} \mid \text{<s>})$	1/3 = 0.33
$P(\text{i} \mid \text{sam})$	1/3 = 0.33
$P(\text{</s>} \mid \text{am})$	1/3 = 0.33
$P(\text{not} \mid \text{am})$	1/3 = 0.33
$P(\text{sam} \mid \text{not})$	1/1 = 1

Sentence	Bigrams Probability
P(<s> I am Sam </s>)	0.148137
P(<s> Sam I am </s>)	0.035937
P(<s> I am not Sam </s>)	0.148137

N-grams: Bigrams probability table

- Estimate N-grams probability
  - Unigrams counting

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

i	want	to	eat	chinese	food	lunch	spend
2533	927	2417	746	158	1093	341	278

N-grams: Bigrams probability table (cont.)

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

- Estimate N-grams probability
  - Bigrams counting (Col given row)
    - “I want” → $count(\text{previous}, \text{current}) = c(w_{i-1}, w_i) = c(\text{I}, \text{want}) = 827$

Current

Previous

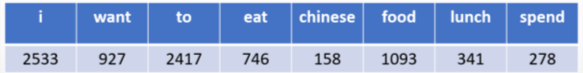
	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

N-grams: Bigrams probability table

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

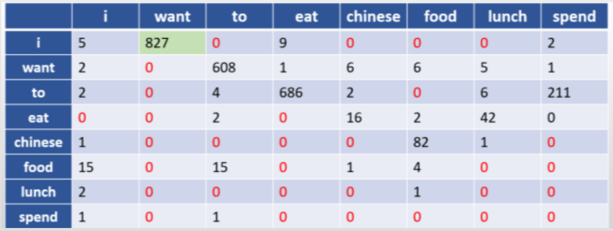
- Estimate N-grams probability
  - Bigrams counting divided by Unigrams counting

Unigrams counting



i	want	to	eat	chinese	food	lunch	spend
2533	927	2417	746	158	1093	341	278

Bigrams counting



	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

Current

Previous

	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054	0.0011
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087
eat	0	0	0.0027	0	0.021	0.0027	0.056	0
chinese	0.0063	0	0	0	0	0.52	0.0063	0
food	0.014	0	0.014	0	0.00092	0.0037	0	0
lunch	0.0059	0	0	0	0	0.0029	0	0
spend	0.0036	0	0.0036	0	0	0	0	0

N-grams: Bigrams probability table

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

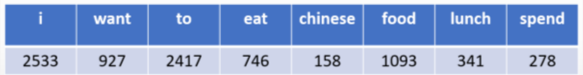
- Estimate N-grams probability
  - Bigrams counting divided by Unigrams counting

$$P(\text{want}|\text{i}) = \frac{count(\text{i,want})}{count(\text{i})} = \frac{827}{2533} = 0.33$$

Current

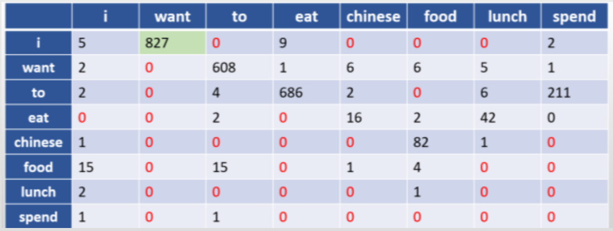
Previous

Unigrams counting



i	want	to	eat	chinese	food	lunch	spend
2533	927	2417	746	158	1093	341	278

Bigrams counting



	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054	0.0011
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087
eat	0	0	0.0027	0	0.021	0.0027	0.056	0
chinese	0.0063	0	0	0	0	0.52	0.0063	0
food	0.014	0	0.014	0	0.00092	0.0037	0	0
lunch	0.0059	0	0	0	0	0.0029	0	0
spend	0.0036	0	0.0036	0	0	0	0	0

N-grams: Bigrams probability table

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

- Estimate N-grams probability
    - Bigrams counting divided by Unigrams counting

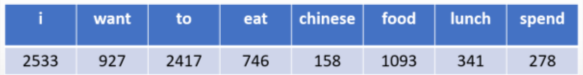
$$P(\text{want}|\text{i}) = \frac{count(\text{i,want})}{count(\text{i})} = \frac{827}{2533} = 0.33$$

Current

Previous

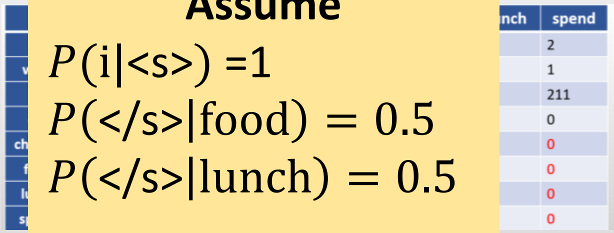
	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054	0.0011
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087
eat	0	0	0.0027	0	0.021	0.0027	0.056	0
chinese	0.0063	0	0	0	0	0.52	0.0063	0
food	0.014	0	0.014	0	0.00092	0.0037	0	0
lunch	0.0059	0	0	0	0	0.0029	0	0
spend	0.0036	0	0.0036	0	0	0	0	0

Unigrams counting



i	want	to	eat	chinese	food	lunch	spend
2533	927	2417	746	158	1093	341	278

Bigrams counting



nch	spend
	2
	1
	211
	0
	0
	0
	0
	0

Assume

$P(\text{i}|\text{<s>})$ =1

$P(\text{</s>}|\text{food}) = 0.5$

$P(\text{</s>}|\text{lunch}) = 0.5$

$P(\text{<s> I eat chinese food </s>}) =$

$P(\text{<s> I spend to lunch </s>}) =$

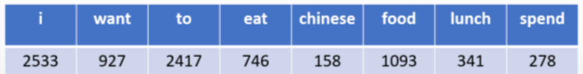
N-grams: Bigrams probability table

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

- Estimate N-grams probability
  - Bigrams counting divided by Unigrams counting

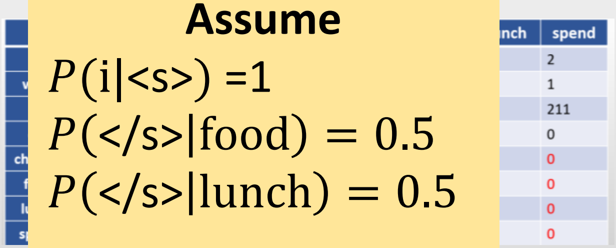
$$P(\text{want}|\text{i}) = \frac{count(\text{i,want})}{count(\text{i})} = \frac{827}{2533} = 0.33$$

Unigrams counting



i	want	to	eat	chinese	food	lunch	spend
2533	927	2417	746	158	1093	341	278

Bigrams counting



	nch	spend
		2
v		1
		211
		0
ch		0
f		0
lu		0
sp		0

Assume

$P(\text{i}|\text{<s>}) = 1$

$P(\text{</s>}|\text{food}) = 0.5$

$P(\text{</s>}|\text{lunch}) = 0.5$

Current

Previous

	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054	0.0011
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087
eat	0	0	0.0027	0	0.021	0.0027	0.056	0
chinese	0.0063	0	0	0	0	0.52	0.0063	0
food	0.014	0	0.014	0	0.00092	0.0037	0	0
lunch	0.0059	0	0	0	0	0.0029	0	0
spend	0.0036	0	0.0036	0	0	0	0	0

$P(\text{<s> I eat chinese food </s>}) = 1 \times 0.0036 \times 0.0021 \times 0.52 \times 0.5 = 1.9 \times 10^{-5}$

$P(\text{<s> I spend to lunch </s>}) = 1 \times 0.00079 \times 0.0036 \times 0.0025 \times 0.5 = 3.5 \times 10^{-9}$

N-grams: Loglikelihood

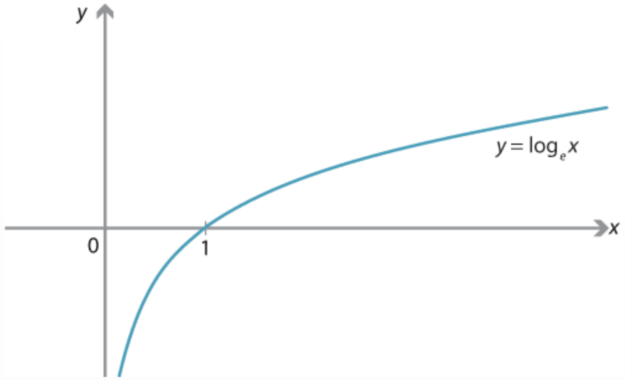
- Do calculating in log space ($\log P(S)$)
    - Avoid underflow (number too small)
    - Adding is faster than multiplying

$$P(A,B,C,D) = P(A)P(B|A)P(C|A,B)P(D|A,B,C)$$

$$\log P(A,B,C,D) = \log P(A)P(B|A)P(C|A,B)P(D|A,B,C)$$

$$= \log P(A) + \log P(B|A) + \log P(C|A,B) + \log P(D|A,B,C)$$

N-grams: Loglikelihood (cont.)



- Calculate log likelihood of the following sentence: “I eat chinese food ”
  - $P($<s> I eat chinese food </s>$)$

$= 1 \times 0.0036 \times 0.0021 \times 0.52 \times 0.5 = 1.9 \times 10^{-5}$

  - $\log P($<s> I eat chinese food </s>$)$

$= \log 1 + \log(0.0036) + \log(0.0021) + \log 0.52 + \log 0.5 = -10.84$

Evaluation

- The model's performance is tested on unseen data
  - Test set
  - Validation set
- Extrinsic Evaluation
  - Measure the performance of a downstream task, e.g., spelling correction, machine translation, etc.
  - Cons: Time-consuming
- Intrinsic Evaluation
  - Evaluate the performance of LM on test set → **Perplexity**
  - Cons: dose not guarantee an improvement of a downstream task, but perplexity often correlates with such improvements

Perplexity

- Perplexity is a quick evaluation metric for language model
- Perplexity can be seen a normalized version of the probability of the test set.
- Perplexity is the inverse probability, normalized by the number of words
- Lower perplexity is better!

$$\mathrm{PP}(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_1, \dots, w_{i-1})}}$$

Perplexity (cont.)

- Perplexity is a quick evaluation metric for language model
- Perplexity can be seen a normalized version of the probability of the test set.
- Perplexity is the inverse probability, normalized by the number of words
- Lower perplexity is better!

$$\mathrm{PP}(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_1, \ldots, w_{i-1})}}$$

Logarithmic version

$$b^{-\frac{1}{N}\sum_{i=1}^{N} \log_b(P(w_i|w_1,\ldots,w_{i-1}))}$$

Perplexity (cont.)

- Perplexity as branching factor:
  - Number of possible next words that can follow any word
- Consider the task of recognizing a string of random digits of length N, given that each of the 10 digits (0-9) occurs with qual probability.

$$\mathrm{PP}(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_1, \ldots, w_{i-1})}}$$

$$= \left(\prod_{i=1}^{N} \frac{1}{P(w_i)}\right)^{\frac{1}{N}} = \left(\prod_{i=1}^{N} \frac{1}{(1/10)}\right)^{\frac{1}{N}} = 10$$

Perplexity (cont.)

- Example: $\mathrm{PP}(W)$ of “I eat chinese food”
  - $\mathrm{PP}(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_1,\ldots,w_{i-1})}}$ or $e^{-\frac{1}{N}\sum_{i=1}^{N} \log_e(P(w_i|w_1,\ldots,w_{i-1}))}$
  - $\mathrm{PP}(\text{<s> I eat chinese food </s>}) = e^{-\frac{1}{5}(\ln 1 \times \ln 0.0036 \times \ln 0.0021 \times \ln 0.52 \times \ln 0.5)} = 8.74$

	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054	0.0011
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087
eat	0	0	0.0027	0	0.021	0.0027	0.056	0
chinese	0.0063	0	0	0	0	0.52	0.0063	0
food	0.014	0	0.014	0	0.00092	0.0037	0	0
lunch	0.0059	0	0	0	0	0.0029	0	0
spend	0.0036	0	0.0036	0	0	0	0	0

Zeros and Unknown words

- Zeros
  - Some probabilities do not occur in the training set but occur in the test set.
  - It still in vocab list.

- Example
  - Training :
    - <s> I am a student . </s>
    - <s> I live in Bangkok . </s>
    - <s> I like to read . </s>
  - Test :
    - <s> I am a teacher . </s>
  - What is the probability of $P(\text{<s> I am a teacher . </s>})$?

    Probability = 0

    - $P(\text{<s>}) \times P(\text{i}|\text{<s>}) \times P(\text{am}|\text{<s> I}) \times P(a|\text{<s> I am}) \times \boxed{P(\text{teacher}|\text{<s> I am a})} \times P(.|\text{<s> I am a teacher}) \times P(\text{</s>}|\text{<s> I am a teacher .})$

Zeros and Unknown words (cont.)

- Zeros
  - Some probabilities do not occur in th
  - It still in vocab list.

- Example
  - Training :
    - <s> I am a student . </s>
    - <s> I live in Bangkok . </s>
    - <s> I like to read . </s>
  - Test :
    - <s> I am a teacher . </s>
  - What is the probability of $P(\text{<s> I am a teacher . </s>})$?
    - $P(\text{<s>}) \times P(\mathrm{i}|\text{<s>}) \times P(\mathrm{am}|\text{<s> I}) \times P(a|\text{<s> I am}) \times \boxed{P(\text{teacher}|\text{<s> I am a})} \times P(.|\text{<s> I am a teacher}) \times P(\text{</s>}|\text{<s> I am a teacher .})$

Probability = 0

- ○ $P(\text{teacher}|\text{<s> I am a}) = 0$
- ○ N-grams with zero probability mean that we will assign 0 probability to the test set.
- ○ We cannot compute **perplexity** (division by 0).

Zeros and Unknown words (cont.)

- Unknown words (UNK) or Out of vocabulary (OOV)
  1. Assign it as a probability of normal word
     - Create a set of vocabulary with **minimum frequency threshold**
       - That is fixed in advanced
       - Or form top n frequency
       - Or words that have frequency more than 1,2,…,v
     - Convert any words in training and testing that is not in this predefined set
       - to ‘UNK’ token.
       - Simply, deal with UNK word as a normal word
  2. Or just define probability of UNK word with constant value

$$P(UNK) = \frac{wc(UNK_{freq=1})}{wc(total)} = \frac{200}{1000} = 0.2$$

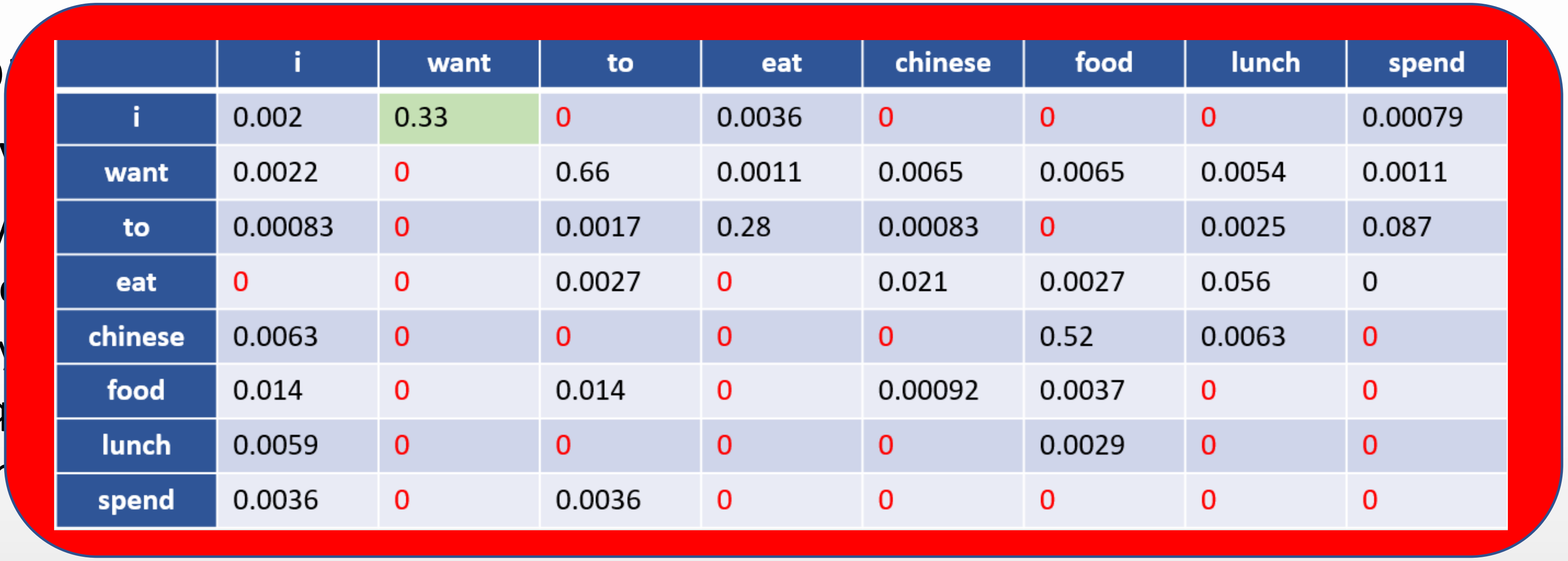
$$P(UNK) = \frac{1}{total\ vocab} = \frac{1}{100} = 0.01$$

Zeros and Unknown words (cont.)

- Unknown words (UNK) o
  1. Assign it as a probabilit
     - Create a set of vocabulary
       - That is fixed in advance
       - Or form top n frequency
       - Or words that have freq
     - Convert any words in trai
       - to 'UNK' token.
       - Simply, deal with UNK word as a normal word
  2. Or just define probability of UNK word with constant value

$$P(UNK) = \frac{1}{total\ vocab} = \frac{1}{100} = 0.01$$

We still have zero problem

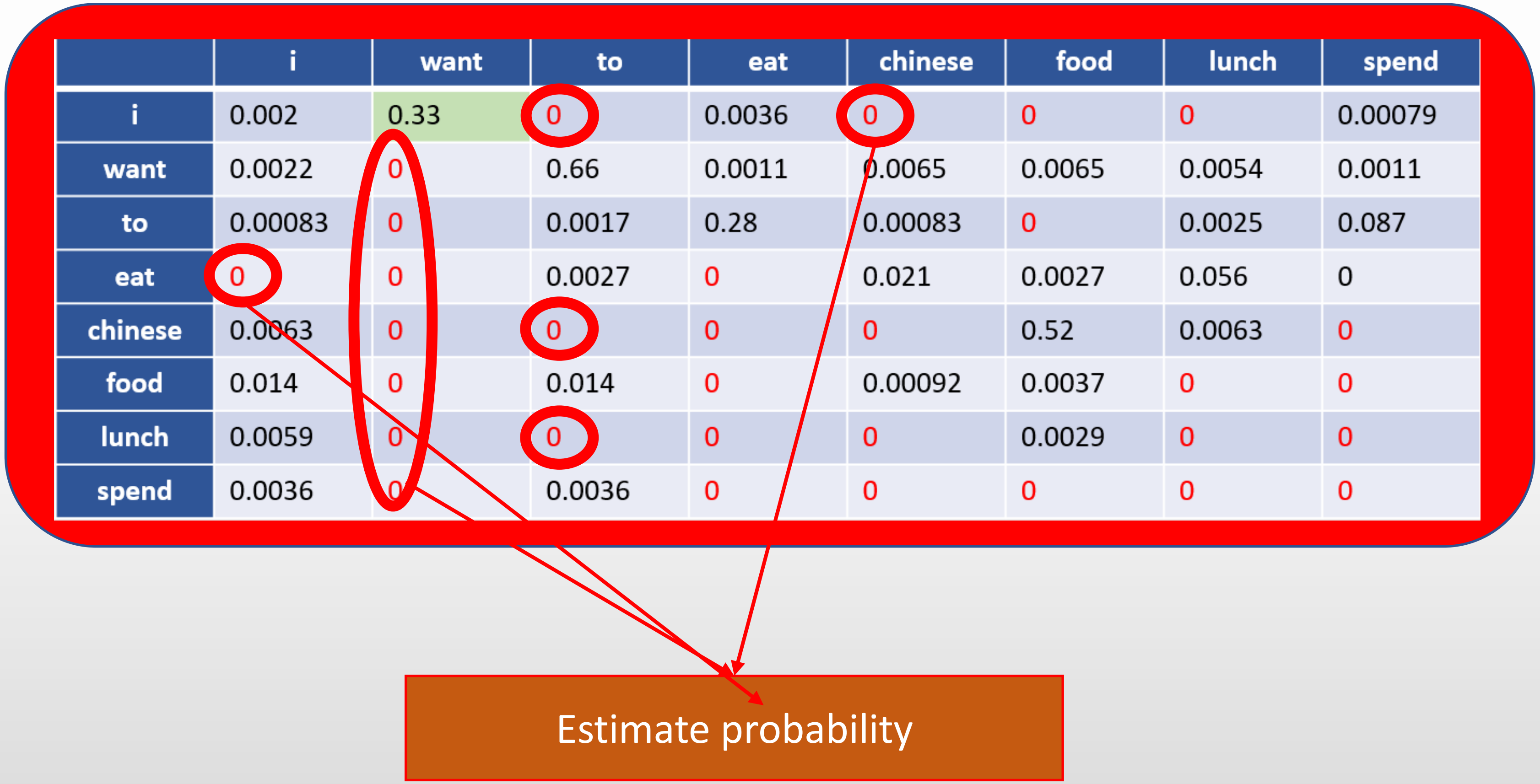


	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054	0.0011
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087
eat	0	0	0.0027	0	0.021	0.0027	0.056	0
chinese	0.0063	0	0	0	0	0.52	0.0063	0
food	0.014	0	0.014	0	0.00092	0.0037	0	0
lunch	0.0059	0	0	0	0	0.0029	0	0
spend	0.0036	0	0.0036	0	0	0	0	0

Smoothing

- Smoothing techniques
  - Add-one estimation
    - OK for text classification, not for LM
  - Back-off
    - For very large N-grams like the Web
  - Interpolation
    - The most commonly used method
  - Kneser-Ney Smoothing
    - The best method

We still have zero problem



	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054	0.0011
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087
eat	0	0	0.0027	0	0.021	0.0027	0.056	0
chinese	0.0063	0	0	0	0	0.52	0.0063	0
food	0.014	0	0.014	0	0.00092	0.0037	0	0
lunch	0.0059	0	0	0	0	0.0029	0	0
spend	0.0036	0	0.0036	0	0	0	0	0

Estimate probability

Smoothing: Add-one estimation

- Add to all the n-grams counts
- For bigram where V is the number of unique word in the corpus

$$P(w_i|w_{i-1}) = \frac{count(w_{i-1}, w_i) + 1}{count(w_{i-1}) + V}$$

	i	want	to	eat	chinese	food	lunch	spend
i	5 +1	827 +1	0 +1	9 +1	0 +1	0 +1	0 +1	2 +1
want	2 +1	0 +1	608 +1	1 +1	6 +1	6 +1	5 +1	1 +1
to	2 +1	0 +1	4 +1	686 +1	2 +1	0 +1	6 +1	211 +1
eat	0 +1	0 +1	2 +1	0 +1	16 +1	2 +1	42 +1	0 +1
chinese	1 +1	0 +1	0 +1	0 +1	0 +1	82 +1	1 +1	0 +1
food	15 +1	0 +1	15 +1	0 +1	1 +1	4 +1	0 +1	0 +1
lunch	2 +1	0 +1	0 +1	0 +1	0 +1	1 +1	0 +1	0 +1
spend	1 +1	0 +1	1 +1	0 +1	0 +1	0 +1	0 +1	0 +1

Smoothing: Add-one estimation (cont.)

- Add to all the n-grams counts
- For bigram where V is the number of unique word in the corpus

$$P(w_i|w_{i-1}) = \frac{count(w_{i-1}, w_i) + 1}{count(w_{i-1}) + V}$$

	i	want	to	eat	chinese	foo		
i	5 +1	827 +1	0 +1	9 +1	0 +1	0 +1		
want	2 +1	0 +1	608 +1	1 +1	6 +1	6 +1		
to	2 +1	0 +1	4 +1	686 +1	2 +1	0 +1		
eat	0 +1	0 +1	2 +1	0 +1	16 +1	2 +1		
chinese	1 +1	0 +1	0 +1	0 +1	0 +1	82 +1	1 +1	0 +1
food	15 +1	0 +1	15 +1	0 +1	1 +1	4 +1	0 +1	0 +1
lunch	2 +1	0 +1	0 +1	0 +1	0 +1	1 +1	0 +1	0 +1
spend	1 +1	0 +1	1 +1	0 +1	0 +1	0 +1	0 +1	0 +1

❑ Pros: Easiest to implement

❑ Cons:
  - Usually perform poorly compare to other techniques.
  - The probabilities change a lot if there are too many zeros n-grams

Smoothing: Back off

- Use less context for contexts you don’t know about
- Back off
  - Trigrams > Bigrams > Unigrams
  - Continue until we get some counts
  - Example:
    - $P(\text{teacher}|\text{<s> I am a}) \sim P(\text{teacher}|\text{ I am a}) \sim P(\text{teacher}|\text{am a}) \sim P(\text{teacher}|\text{a}) \sim P(\text{teacher}) \sim P(UNK)$

Smoothing: Interpolation

- Mix unigram, bigram, trigram

$$\hat{P}\ (w_i|w_{i-2}, w_{i-1}) = \lambda_3 P(w_i|w_{i-2}, w_{i-1}) + \lambda_2 P(w_i|w_{i-1}) + \lambda_1 P(w_i) + \lambda_0 C$$

- λ is chosen from testing on validating data set, and $\sum_i \lambda = 1$
- C is (1/vocab) in corpus

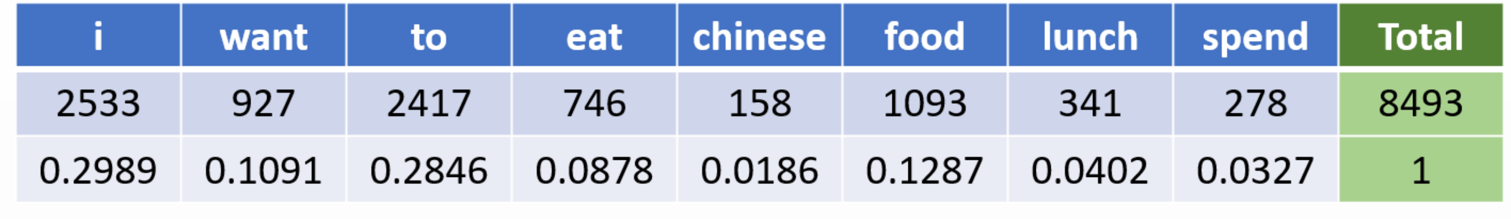
Smoothing: Interpolation (cont.)

- Interpolation for bigrams

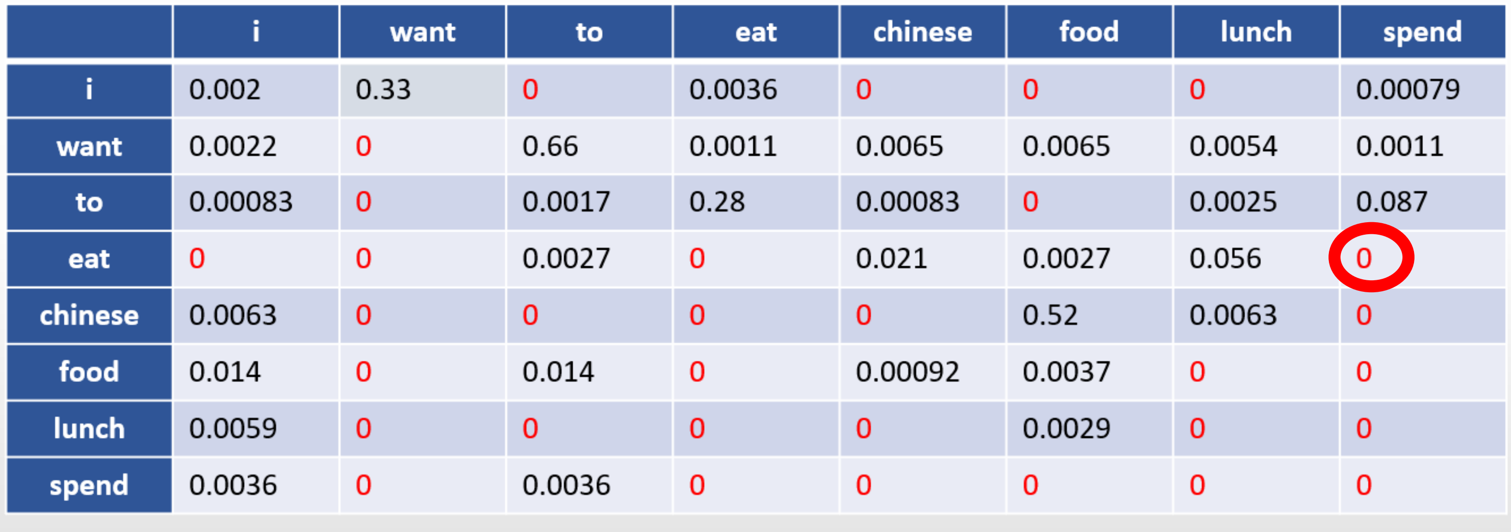
$$\hat{P}\ (w_i|w_{i-2}\ ) = \lambda_2 P(w_i|w_{i-1}) + \lambda_1 P(w_i) + \lambda_0 C$$

#Vocab in corpus = 1446

$P(\text{eat spend}) = P(\text{spend}|\text{eat}) = ?$



i	want	to	eat	chinese	food	lunch	spend	Total
2533	927	2417	746	158	1093	341	278	8493
0.2989	0.1091	0.2846	0.0878	0.0186	0.1287	0.0402	0.0327	1

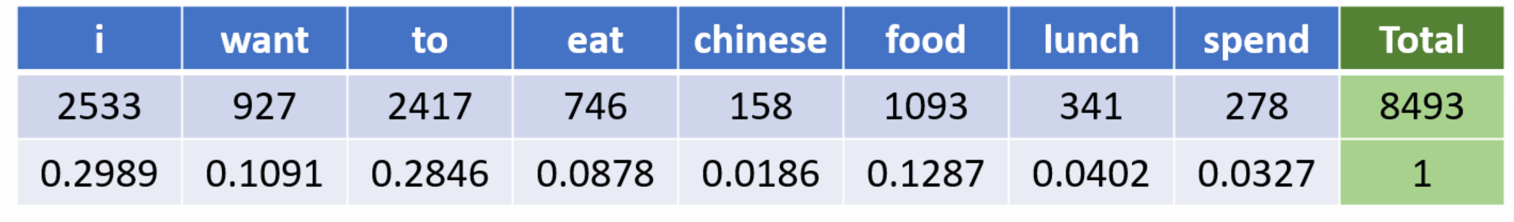


	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054	0.0011
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087
eat	0	0	0.0027	0	0.021	0.0027	0.056	0
chinese	0.0063	0	0	0	0	0.52	0.0063	0
food	0.014	0	0.014	0	0.00092	0.0037	0	0
lunch	0.0059	0	0	0	0	0.0029	0	0
spend	0.0036	0	0.0036	0	0	0	0	0

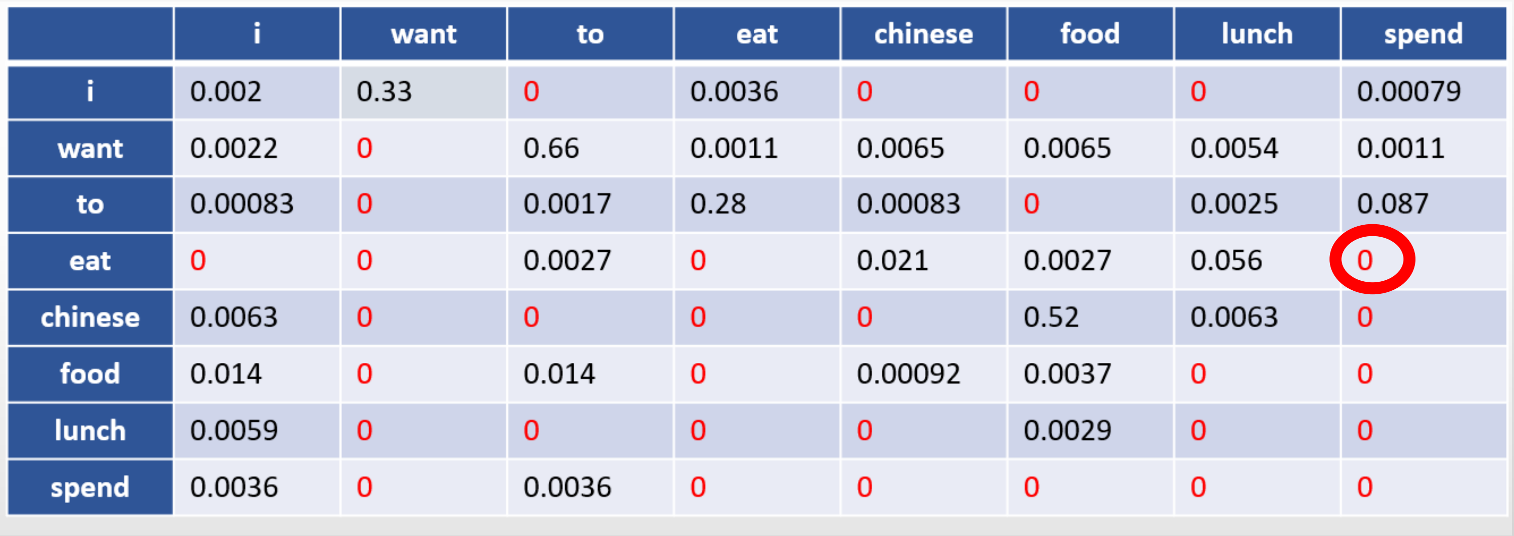
Smoothing: Interpolation (cont.)

- Interpolation for bigrams

$$\hat{P}\ (w_i|w_{i-2}\ ) = \lambda_2 P(w_i|w_{i-1}) + \lambda_1 P(w_i) + \lambda_0 C$$



i	want	to	eat	chinese	food	lunch	spend	Total
2533	927	2417	746	158	1093	341	278	8493
0.2989	0.1091	0.2846	0.0878	0.0186	0.1287	0.0402	0.0327	1



	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054	0.0011
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087
eat	0	0	0.0027	0	0.021	0.0027	0.056	0
chinese	0.0063	0	0	0	0	0.52	0.0063	0
food	0.014	0	0.014	0	0.00092	0.0037	0	0
lunch	0.0059	0	0	0	0	0.0029	0	0
spend	0.0036	0	0.0036	0	0	0	0	0

#Vocab in corpus = 1446

$$P(\text{eat spend}) = P(\text{spend}|\text{eat})$$

$$= \lambda_2 P(\text{spend}|\text{eat}) + \lambda_1 P(\text{spend}) +$$

$$\lambda_0 C$$

$$= 0.7 \times 0 + 0.25 \times 0.0327 +$$

$$0.05 \times (1/1446)$$

$$= 0.00820958$$

Smoothing: Kneser-Ney Smoothing

- Absolute discounting: save some probability mass for the zeros
  - Church and Gale (1991)
    - AP newswire dataset
    - 22 million words in training set
    - Next 22 million words in validation set
  - Suppose we want to subtract little form a count of 4 to save probability mass for the zeros
    - How much to subtract
  - On average, a bigram that occurred **4 times** in the first 22 million words (training) occurred **3.23 times** in the next 22 million words (validation)
    - The discrepancy between train and validate is 4 – 3.23 = 0.77
    - The averaging discrepancy of all words in about **0.75 (called discount, d)**

Bigram count in training	Bigram count in validating set
0	0.0000270
1	0.448
2	1.25 (~-0.75)
3	2.24 (~-0.75)
4	3.23 (~-0.75)
5	4.21 (~-0.75)
6	5.23 (~-0.75)
7	6.21 (~-0.75)
8	7.21 (~-0.75)
9	8.26 (~-0.75)

Smoothing: Kneser-Ney Smoothing (cont.)

- Absolute discounting formalizers this intuition by subtracting a fixed (absolute) d = 0.75 from each count.

Discount bigram

Unigram

$$P_{\mathrm{absoluteDiscounting}}(w_{\mathrm{i}}|w_{i-1}) = \frac{c(w_{i-1}, w_i) - d}{c(w_{i-1})} + \lambda(w_{i-1})P(w_i)$$

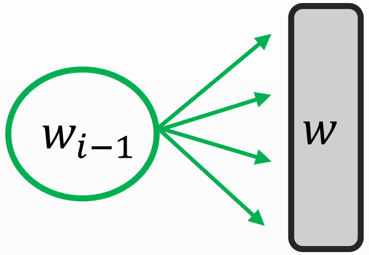
Interpolation weight

- But should we just use the regular unigram?
  - Solution: Kneser-Ney Smoothing

Bigram count in training	Bigram count in validating set
0	0.0000270
1	0.448
2	1.25 (~-0.75)
3	2.24 (~-0.75)
4	3.23 (~-0.75)
5	4.21 (~-0.75)
6	5.23 (~-0.75)
7	6.21 (~-0.75)
8	7.21 (~-0.75)
9	8.26 (~-0.75)

Smoothing: Kneser-Ney Smoothing (cont.)

- Kneser-Ney smoothing for bigram



$P(w_{i-1}, w_i) \approx$

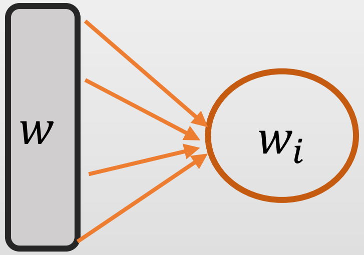
$$P_{KN}(w_i|w_{i-1}) = \frac{\max(count(w_{i-1}, w_i) - d, 0)}{count(w_{i-1})} + \lambda(w_{i-1})P_{continuation}(w_i)$$

Other continues from $P(w_{i-1})$

$P(w_i)$ continues from others

$$P_{continuation}(w_i) = \frac{|\{w_{i-1}:count(w_{i-1},w)>0\}|}{\sum_{w'} |\{w'_{i-1}:count(w'_{i-1},w')>0\}|}$$

$$\lambda(w_{i-1}) = d\frac{|\{w: c(w_{i-1}, w) > 0\}|}{c(w_{i-1})}$$

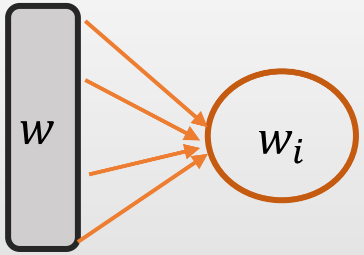


Smoothing: Kneser-Ney Smoothing (cont.)

$$P_{continuation}(w_i) = \frac{|\{w_{i-1}:count(w_{i-1},w)>0\}|}{\sum_{w'}|\{w'_{i-1}:count(w'_{i-1},w')>0\}|}$$

$P(w_i)$ continues from others

- How to calculate $P_{continuation}(w)$
  - Example:
  - If our corpus contain these bigrams
    - {San Francisco, San Francisco, San Francisco, Sun glasses, Reading glasses, Colored glasses}
  - $P(\text{Francisco}) =$
  - $P(\text{glasses}) =$
  - $P_{continuation}(\text{Francisco}) =$
  - $P_{continuation}(\text{glasses}) =$



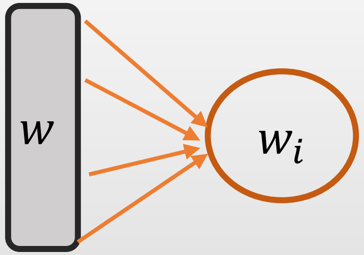
Smoothing: Kneser-Ney Smoothing (cont.)

- How to calculate $P_{continuation}(w)$
  - Example:
  - If our corpus contain these bigrams
    - {San Francisco, San Francisco, San Francisco, Sun glasses, Reading glasses, Colored glasses}
  - $P(\text{Francisco}) = \frac{3}{6}$
  - $P(\text{glasses}) = \frac{3}{6}$
  - $P_{continuation}(\text{Francisco}) =$
  - $P_{continuation}(\text{glasses}) =$

$$P_{continuation}(w) = \frac{|\{w_{i-1}:count(w_{i-1},w)>0\}|}{\sum_{w'}|\{w'_{i-1}:count(w'_{i-1},w')>0\}|}$$

$P(w)$ continues from others

#Distinct pattern = 4



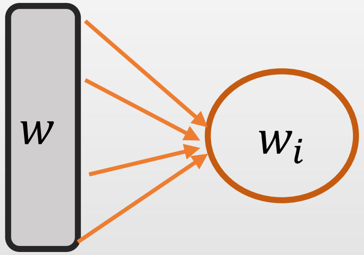
Smoothing: Kneser-Ney Smoothing (cont.)

$$P_{continuation}(w) = \frac{|\{w_{i-1}: count(w_{i-1}, w) > 0\}|}{\sum_{w'} |\{w'_{i-1}: count(w'_{i-1}, w') > 0\}|}$$

$P(w)$ continues from others

- How to calculate $P_{continuation}(w)$
  - Example:
  - If our corpus contain these bigrams
    - {San Francisco, San Francisco, San Francisco, Sun glasses, Reading glasses, Colored glasses}
  - $P(\text{Francisco}) = \frac{3}{6}$
  - $P(\text{glasses}) = \frac{3}{6}$
  - $P_{continuation}(\text{Francisco}) = \frac{1}{4}$
  - $P_{continuation}(\text{glasses}) = \frac{3}{4}$

#Distinct pattern = 4



Smoothing: Kneser-Ney Smoothing (cont.)

- P_{KN} will continue recursively until it reaches unigram.
- Assume trigrams
  - $P_{KN}(\text{Trigram}) = \frac{\max(count(w_{i-2},w_{i-1},w_i)-d,0)}{count(w_{i-2},w_{i-1})} + \lambda(w_{i-1}, w_{i-2})P_{KN}(\text{Bigrams})$
  - $P_{KN}(\text{Bigram}) = \frac{\max(count(w_{i-1},w_i)-d,0)}{count(w_{i-1})} + \lambda(w_{i-1})P_{KN}(\text{Unigrams})$
  - $P_{KN}(\text{Unigram}) = \frac{\max(count(w_i\ )-d,0)}{count(w)} + \frac{1}{V} \quad ; \frac{1}{V} = P(UNK)$

Smoothing: Kneser-Ney Smoothing (cont.)

- Example: We have the following training corpus.

  <s> I am Sam </s>

  <s> Sam I am </s>

  <s> I am Sam </s>

  <s> I like green eggs </s>

- Train a bigram Kneser-Ney model.

Smoothing: Kneser-Ney Smoothing (cont.)

- Create a unigram counting table

<s>	I	am	Sam	like	green	eggs	</s>
4	4	3	3	1	1	1	4

Training corpus:
<s> I am Sam </s>
<s> Sam I am </s>
<s> I am Sam </s>
<s> I like green eggs </s>

Smoothing: Kneser-Ney Smoothing (cont.)

Training corpus:
<s> I am Sam </s>
<s> Sam I am </s>
<s> I am Sam </s>
<s> I like green eggs </s>

- Create a unigram counting table

<s>	I	am	Sam	like	green	eggs	</s>
4	4	3	3	1	1	1	4

- Create a bigram counting table

	<s>	I	am	Sam	like	green	eggs	</s>
<s>	0	3	0	1	0	0	0	0
I	0	0	3	0	1	0	0	0
Am	0	0	0	2	0	0	0	1
Sam	0	1	0	0	0	0	0	2
Like	0	0	0	0	0	1	0	0
green	0	0	0	0	0	0	1	0
Eggs	0	0	0	0	0	0	0	1
</s>	0	0	0	0	0	0	0	0

Smoothing: Kneser-Ney Smoothing (cont.)

- Create a unigram counting table

<s>	I	am	Sam	like	green	eggs	</s>
4	4	3	3	1	1	1	4

- Create a bigram counting table

	<s>	I	am	Sam	like	green	eggs	</s>
<s>	0	3	0	1	0	0	0	0
I	0	0	3	0	1	0	0	0
Am	0	0	0	2	0	0	0	1
Sam	0	1	0	0	0	0	0	2
Like	0	0	0	0	0	1	0	0
green	0	0	0	0	0	0	1	0
Eggs	0	0	0	0	0	0	0	1
</s>	0	0	0	0	0	0	0	0

Training corpus:
<s> I am Sam </s>
<s> Sam I am </s>
<s> I am Sam </s>
<s> I like green eggs </s>

Compute the log-likelihood of the sentence **“<s> am Sam </s>”** using Kneser-Ney Smoothing

$$LL(P(\text{<s> am Sam </s>})) =$$
$$\ln\big(P(\text{am}|\text{<s>})\big) + \ln\big(P(\text{Sam}|\text{am})\big) + \ln\big(P(\text{</s>}|\text{Sam})\big)$$

Smoothing: Kneser-Ney Smoothing (cont.)

- Create a unigram counting table

<s>	I	am	Sam	like	green	eggs	</s>
4	4	3	3	1	1	1	4

- Create a bigram counting table

	<s>	I	am	Sam	like	green	eggs	</s>
<s>	0	3	0	1	0	0	0	0
I	0	0	3	0	1	0	0	0
am	0	0	0	2	0	0	0	1
Sam	0	1	0	0	0	0	0	2
Like	0	0	0	0	0	1	0	0
green	0	0	0	0	0	0	1	0
Eggs	0	0	0	0	0	0	0	1
</s>	0	0	0	0	0	0	0	0

Training corpus:
<s> I am Sam </s>
<s> Sam I am </s>
<s> I am Sam </s>
<s> I like green eggs </s>

Compute the log-likelihood of the sentence **"<s> am Sam </s>"** using Kneser-Ney Smoothing

$$LL(P(\text{<s> am Sam </s>})) = \ln\big(P(\text{am}|\text{<s>})\big) + \ln\big(P(\text{Sam}|\text{am})\big) + \ln\big(P(\text{</s>}|\text{Sam})\big)$$

$$P(\text{Sam}|\text{am}) = P_{kn}(Sam|am) = \left(\frac{\max(2-0.75,0)}{3}\right) + \left(0.75 \times \frac{2}{3}\right) \times \frac{2}{11} = \mathbf{0.5076}$$

Smoothing: Kneser-Ney Smoothing (cont.)

- Create a unigram counting table

<s>	I	am	Sam	like	green	eggs	</s>
4	4	3	3	1	1	1	4

- Create a bigram counting table

	<s>	I	am	Sam	like	green	eggs	</s>
<s>	0	3	0	1	0	0	0	0
I	0	0	3	0	1	0	0	0
am	0	0	0	2	0	0	0	1
Sam	0	1	0	0	0	0	0	2
Like	0	0	0	0	0	1	0	0
green	0	0	0	0	0	0	1	0
Eggs	0	0	0	0	0	0	0	1
</s>	0	0	0	0	0	0	0	0

Training corpus:
<s> I am Sam </s>
<s> Sam I am </s>
<s> I am Sam </s>
<s> I like green eggs </s>

Compute the log-likelihood of the sentence **"<s> am Sam </s>"** using Kneser-Ney Smoothing

$$LL(P(\text{<s> am Sam </s>})) = \ln\big(P(\text{am}|\text{<s>})\big) + \ln\big(P(\text{Sam}|\text{am})\big) + \ln\big(P(\text{</s>}|\text{Sam})\big)$$

$$P(\text{Sam}|\text{am}) = P_{kn}(\text{Sam}|\text{am}) = \boxed{\left(\frac{\max(2-0.75,0)}{3}\right)} + \left(0.75 \times \frac{2}{3}\right) \times \frac{2}{11} = \mathbf{0.5076}$$

Smoothing: Kneser-Ney Smoothing (cont.)

- Create a unigram counting table

<s>	I	am	Sam	like	green	eggs	</s>
4	4	3	3	1	1	1	4

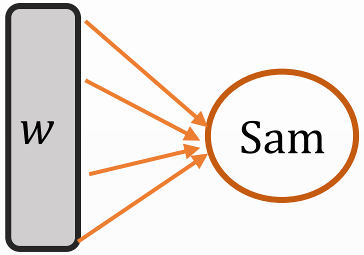
- Create a bigram counting table

	<s>	I	am	Sam	like	green	eggs	</s>
<s>	0	3	0	1	0	0	0	0
I	0	0	3	0	1	0	0	0
am	0	0	0	2	0	0	0	1
Sam	0	1	0	0	0	0	0	2
Like	0	0	0	0	0	1	0	0
green	0	0	0	0	0	0	1	0
Eggs	0	0	0	0	0	0	0	1
</s>	0	0	0	0	0	0	0	0

#Distinct pattern = #Filled box = 11

#Distinct pattern of Sam = #Filled box of Sam = 2

Training corpus:
<s> I am Sam </s>
<s> Sam I am </s>
<s> I am Sam </s>
<s> I like green eggs </s>



Compute the log-likelihood of the sentence **"<s> am Sam </s>"** using Kneser-Ney Smoothing

$$LL(P(\text{<s> am Sam </s>})) = \ln\big(P(\text{am}|\text{<s>})\big) + \ln\big(P(\text{Sam}|\text{am})\big) + \ln\big(P(\text{</s>}|\text{Sam})\big)$$

$$P(\text{Sam}|\text{am}) = P_{kn}(Sam|am) = \left(\frac{\max(2-0.75,0)}{3}\right) + \left(0.75 \times \frac{2}{3}\right) \times \boxed{\frac{2}{11}} = \mathbf{0.5076}$$

Smoothing: Kneser-Ney Smoothing (cont.)

- Create a unigram counting table

<s>	I	am	Sam	like	green	eggs	</s>
4	4	$\boxed{3}$	3	1	1	1	4

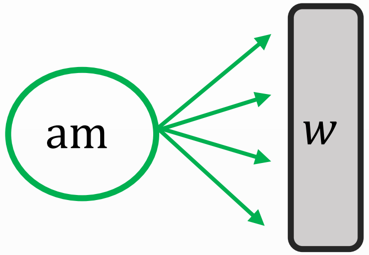
#am= 3

- Create a bigram counting table

	<s>	I	am	Sam	like	green	eggs	</s>
<s>	0	3	0	1	0	0	0	0
I	0	0	3	0	1	0	0	0
am	0	0	0	2	0	0	0	1
Sam	0	1	0	0	0	0	0	2
Like	0	0	0	0	0	1	0	0
green	0	0	0	0	0	0	1	0
Eggs	0	0	0	0	0	0	0	1
</s>	0	0	0	0	0	0	0	0

#Distinct pattern of am = #Filled box of am = 2

Training corpus:
<s> I am Sam </s>
<s> Sam I am </s>
<s> I am Sam </s>
<s> I like green eggs </s>

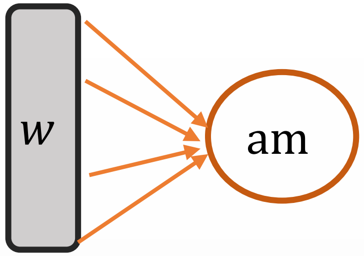


Compute the log-likelihood of the sentence **"<s> am Sam </s>"** using Kneser-Ney Smoothing

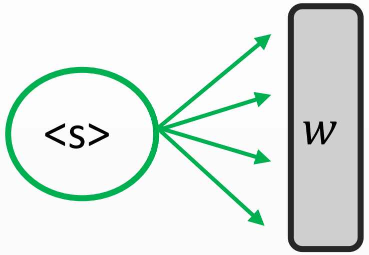
$$LL(P(\text{<s> am Sam </s>})) = \ln\big(P(\text{am}|\text{<s>})\big) + \ln\big(P(\text{Sam}|\text{am})\big) + \ln\big(P(\text{</s>}|\text{Sam})\big)$$

$$P(\text{Sam}|\text{am}) = P_{kn}(Sam|am)$$
$$= \left(\frac{\max(2-0.75,0)}{3}\right) + \left(0.75 \times \boxed{\frac{2}{3}}\right) \times \frac{2}{11} = \mathbf{0.5076}$$

Smoothing: Kneser-Ney Smoothing (cont.)



Training corpus:
<s> I am Sam </s>
<s> Sam I am </s>
<s> I am Sam </s>
<s> I like green eggs </s>



- Create a unigram counting table

<s>	I	am	Sam	like	green	eggs	</s>
4	4	3	3	1	1	1	4

- Create a bigram counting table

	<s>	I	am	Sam	like	green	eggs	</s>
<s>	0	3	0	1	0	0	0	0
I	0	0	3	0	1	0	0	0
am	0	0	0	2	0	0	0	1
Sam	0	1	0	0	0	0	0	2
Like	0	0	0	0	0	1	0	0
green	0	0	0	0	0	0	1	0
Eggs	0	0	0	0	0	0	0	1
</s>	0	0	0	0	0	0	0	0

Compute the log-likelihood of the sentence **"<s> am Sam </s>"** using Kneser-Ney Smoothing

$$LL(P(\text{<s> am Sam </s>})) =$$
$$\ln\big(P(\text{am}|\text{<s>})\big) + \ln\big(P(\text{Sam}|\text{am})\big) + \ln\big(P(\text{</s>}|\text{Sam})\big)$$

$$P(\text{am}|\text{<s>}) = P_{kn}(\text{am}|\text{<s>})$$
$$= \left(\frac{\max(0-0.75,0)}{4}\right) + \left(0.75 \times \frac{2}{4}\right) \times \frac{1}{11} = \mathbf{0.03409}$$

Smoothing: Kneser-Ney Smoothing (cont.)

- Create a unigram counting table

<s>	I	am	Sam	like	green	eggs	</s>
4	4	3	3	1	1	1	4

- Create a bigram counting table

	<s>	I	am	Sam	like	green	eggs	</s>
<s>	0	3	0	1	0	0	0	0
I	0	0	3	0	1	0	0	0
am	0	0	0	2	0	0	0	1
Sam	0	1	0	0	0	0	0	2
Like	0	0	0	0	0	1	0	0
green	0	0	0	0	0	0	1	0
Eggs	0	0	0	0	0	0	0	1
</s>	0	0	0	0	0	0	0	0

Training corpus:
<s> I am Sam </s>
<s> Sam I am </s>
<s> I am Sam </s>
<s> I like green eggs </s>

Compute the log-likelihood of the sentence **“<s> am Sam </s>”** using Kneser-Ney Smoothing

$$LL(P(\text{<s> am Sam </s>})) = \ln\big(P(\text{am}|\text{<s>})\big) + \ln\big(P(\text{Sam}|\text{am})\big) + \ln\big(P(\text{</s>}|\text{Sam})\big)$$

$$P(\text{</s>}|\text{Sam}) = P_{kn}(\text{</s>}|\text{Sam})$$
$$= \left(\frac{\max(2-0.75,0)}{3}\right) + \left(0.75 \times \frac{2}{3}\right) \times \frac{3}{11} = \mathbf{0.5530}$$

Smoothing: Kneser-Ney Smoothing (cont.)

- Create a unigram counting table

<s>	I	am	Sam	like	green	eggs	</s>
4	4	3	3	1	1	1	4

- Create a bigram counting table

	<s>	I	am	Sam	like	green	eggs	</s>
<s>	0	3	0	1	0	0	0	0
I	0	0	3	0	1	0	0	0
Am	0	0	0	2	0	0	0	1
Sam	0	1	0	0	0	0	0	2
Like	0	0	0	0	0	1	0	0
green	0	0	0	0	0	0	1	0
Eggs	0	0	0	0	0	0	0	1
</s>	0	0	0	0	0	0	0	0

Training corpus:
<s> I am Sam </s>
<s> Sam I am </s>
<s> I am Sam </s>
<s> I like green eggs </s>

Compute the log-likelihood of the sentence **"<s> am Sam </s>"** using Kneser-Ney Smoothing

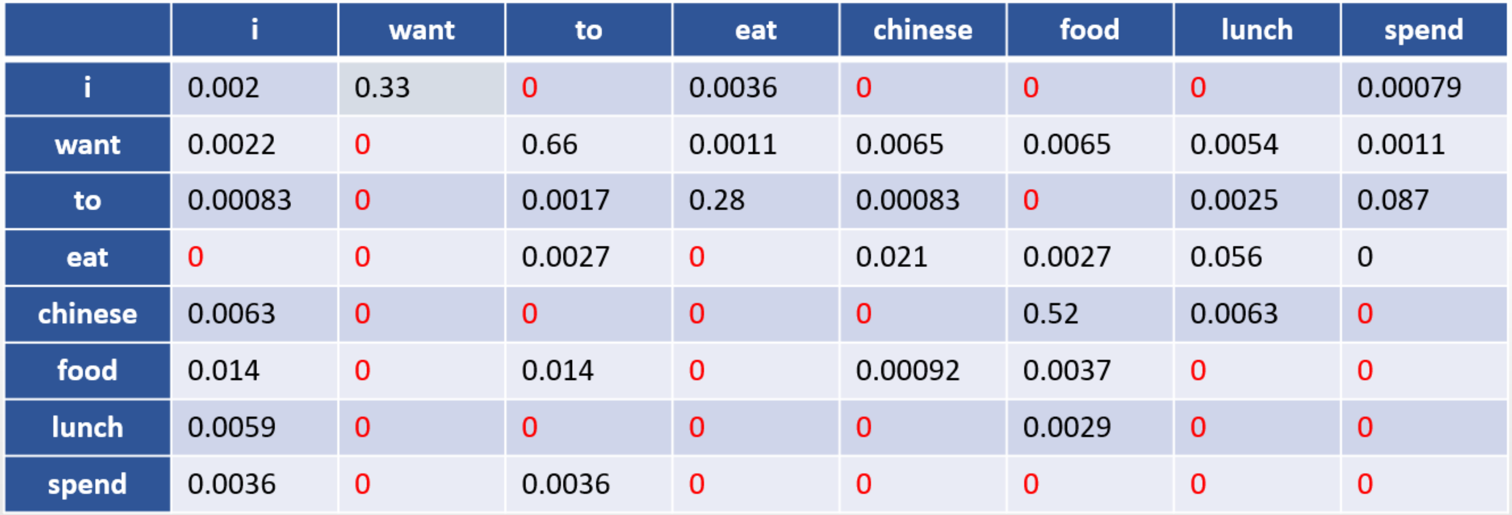
$$LL(P(\text{<s> am Sam </s>})) =$$
$$\ln\big(P(\text{am}|\text{<s>})\big) + \ln\big(P(\text{Sam}|\text{am})\big) + \ln\big(P(\text{</s>}|\text{Sam})\big)$$

$$= \ln(\mathbf{0.03409}) + \ln(\mathbf{0.5076}) + \ln(\mathbf{0.5530})$$

$$= \mathbf{-4.6492}$$

Neural Language Model

- Traditional language model
  - Need a lot of memory to store all those n-grams

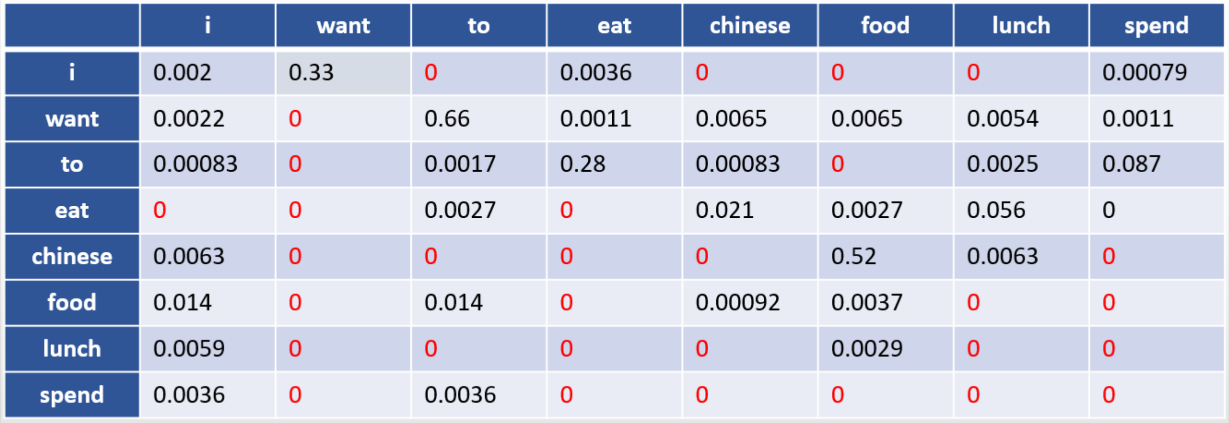


	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054	0.0011
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087
eat	0	0	0.0027	0	0.021	0.0027	0.056	0
chinese	0.0063	0	0	0	0	0.52	0.0063	0
food	0.014	0	0.014	0	0.00092	0.0037	0	0
lunch	0.0059	0	0	0	0	0.0029	0	0
spend	0.0036	0	0.0036	0	0	0	0	0

  - It lacks long-term dependency
    - Example: “ Jane walked into the room. John waked in too. It was late in the day, and everyone was walking home after a long day work. Jane said hi to ______

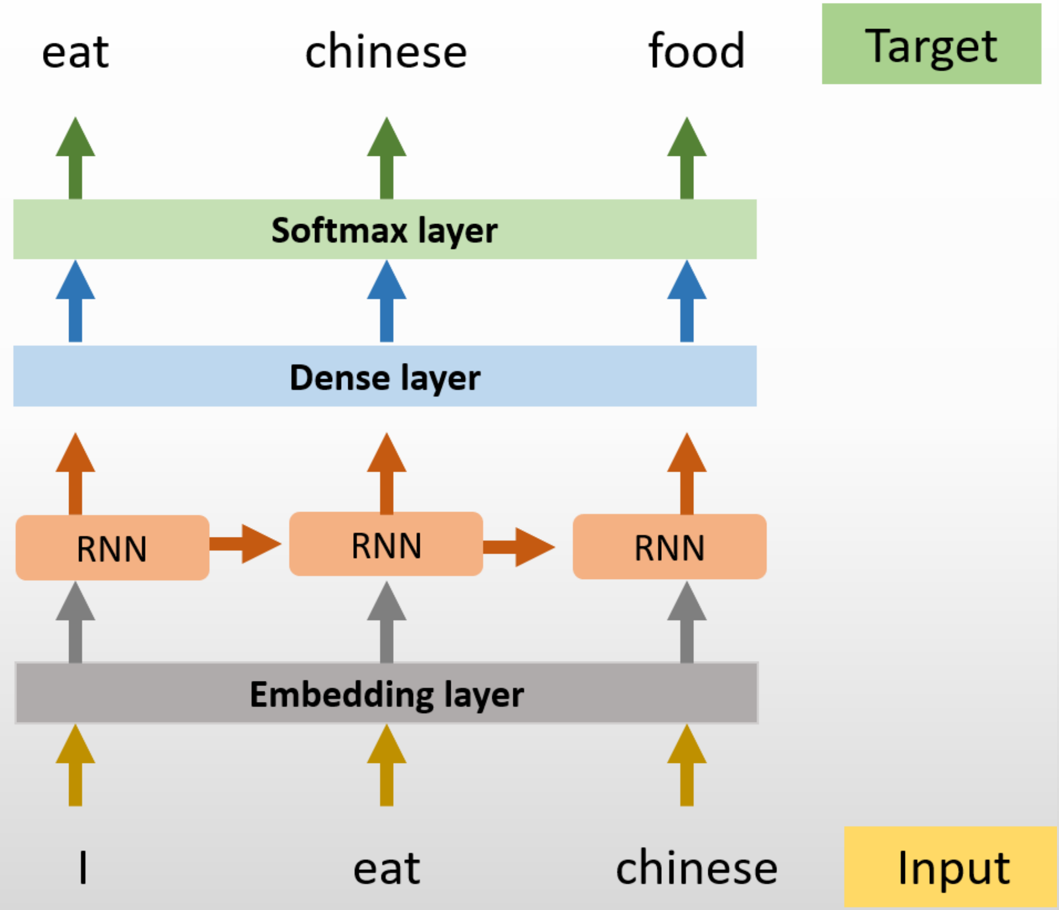
Neural Language Model (cont.)

- Recurrent Neural Network
  - Consider all previous words in the corpus
  - In LM
    - Input(x) is current word in vector form
    - Output(y) is the next word
  - Usually, RNN’s performance is better than traditional language model



	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054	0.0011
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087
eat	0	0	0.0027	0	0.021	0.0027	0.056	0
chinese	0.0063	0	0	0	0	0.52	0.0063	0
food	0.014	0	0.014	0	0.00092	0.0037	0	0
lunch	0.0059	0	0	0	0	0.0029	0	0
spend	0.0036	0	0.0036	0	0	0	0	0

I eat chinese food



Neural Language Model (cont.)

- Recurrent Neural Network

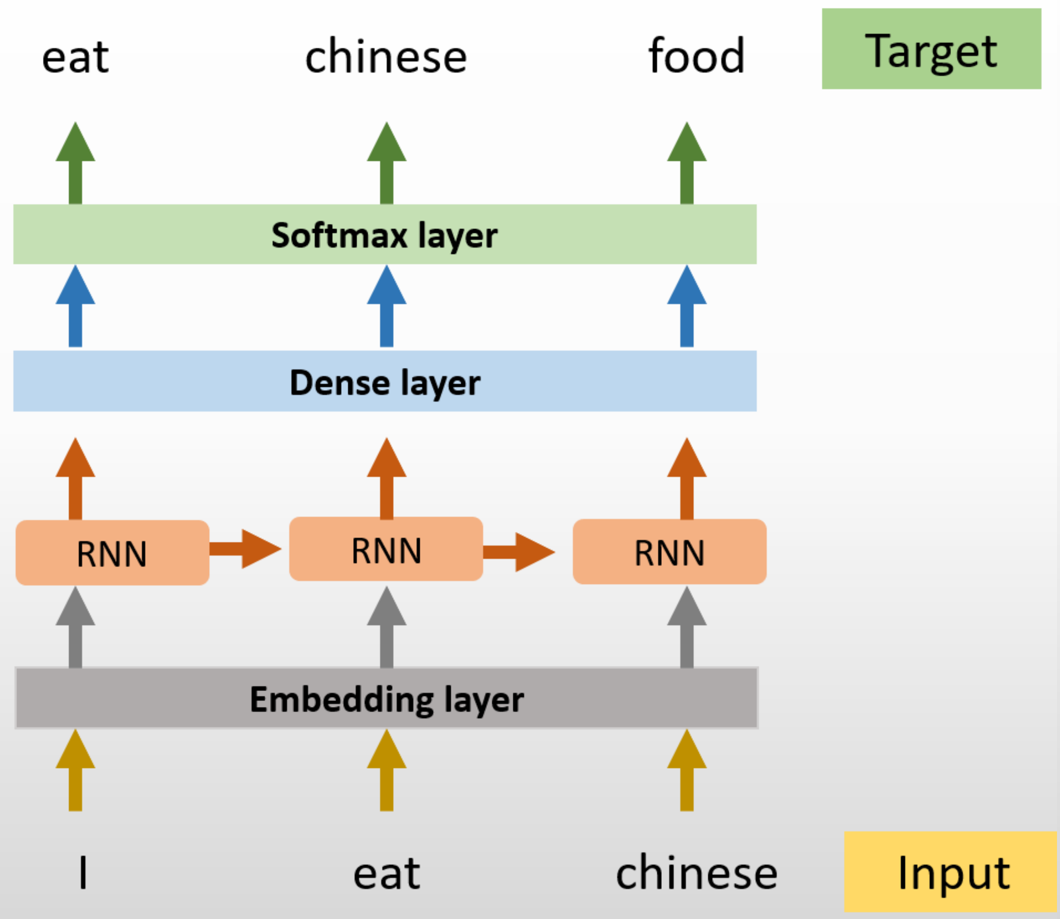
➢ RNN suffer from vanishing gradient

- Long-Short Term Memory
- Gate Recurrent Unit

➢ About Bidirectional RNN

- Bidirectional RNN cannot apply here since we predict the next word and cannot use the future information (violating assumption)

➢ N-gram is still quite useful and often are incorporated to neural language models



Neural Language Model (cont.)

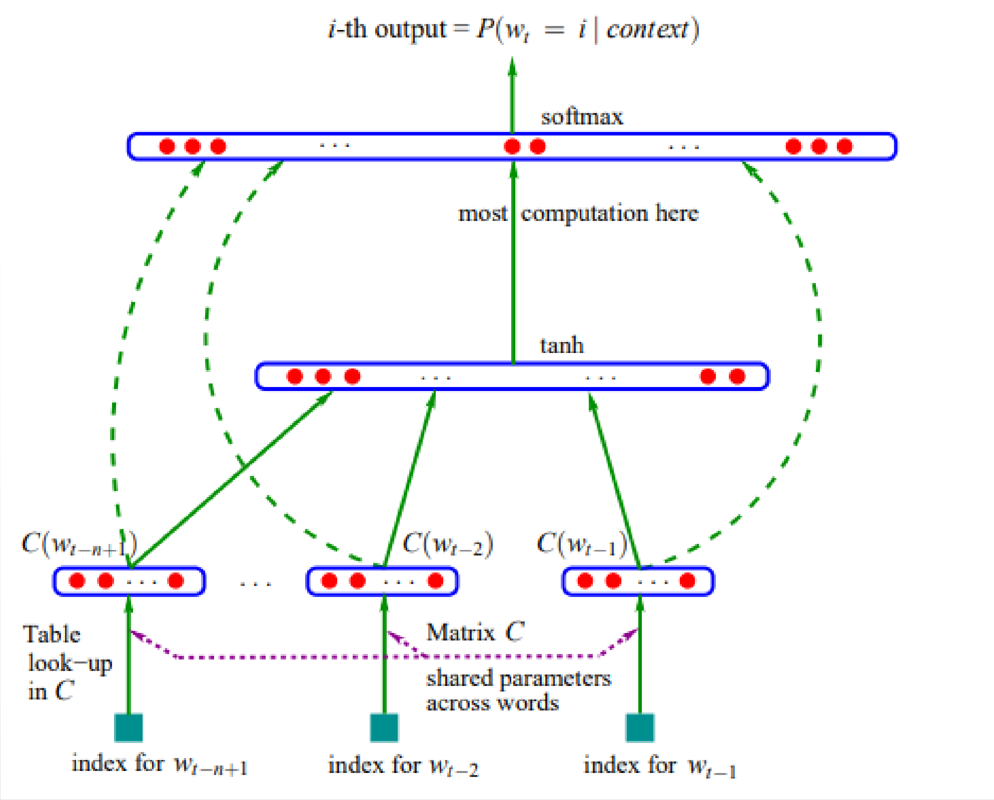
- Cost function

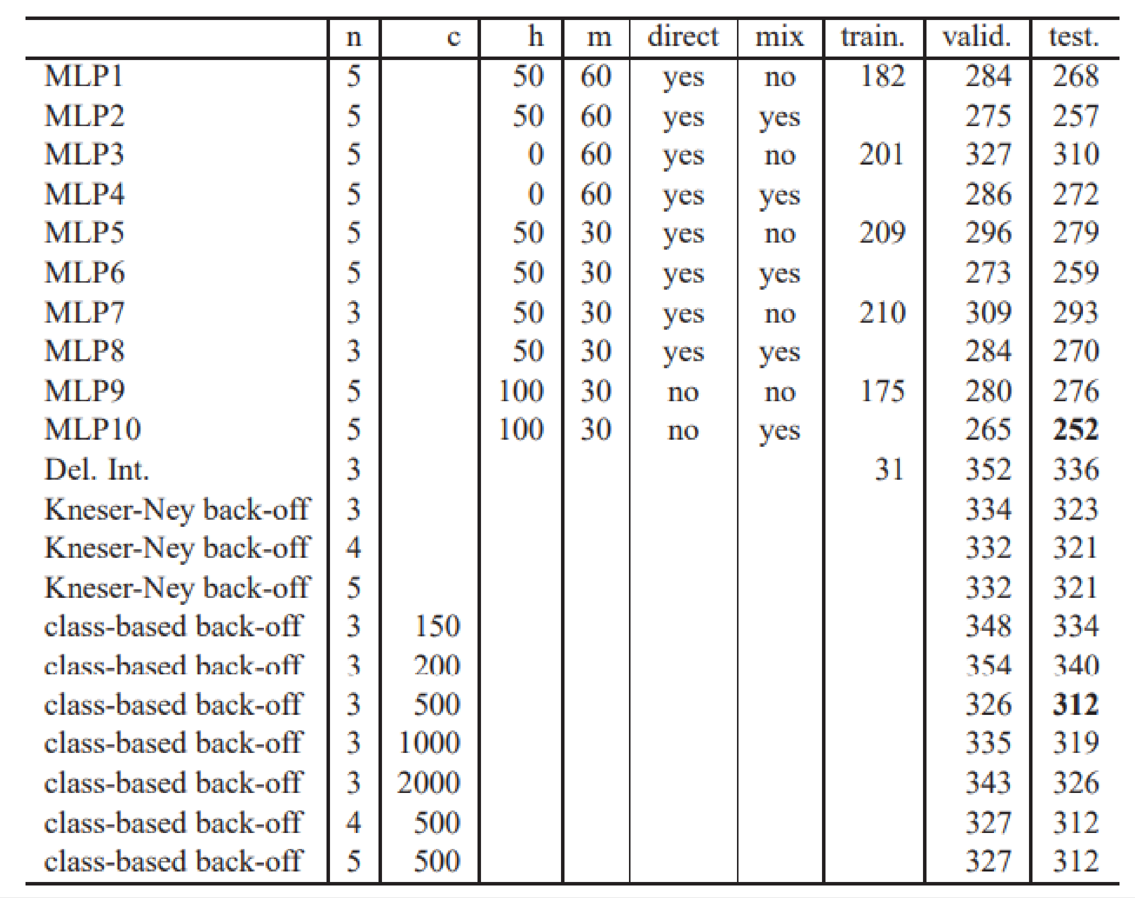
$$J = -\frac{1}{T}\sum_{t=1}^{T}\sum_{j=1}^{|V|} y_{t,j} \log \hat{y}_{t,j}$$

  - where
    - V = Number of unique words in corpus
    - T = Number of total word in corpus
    - y = Target next word
    - $\hat{y}$ = Distribution of predicted next word

- Perplexity = e^J

Neural Language Model (cont.)



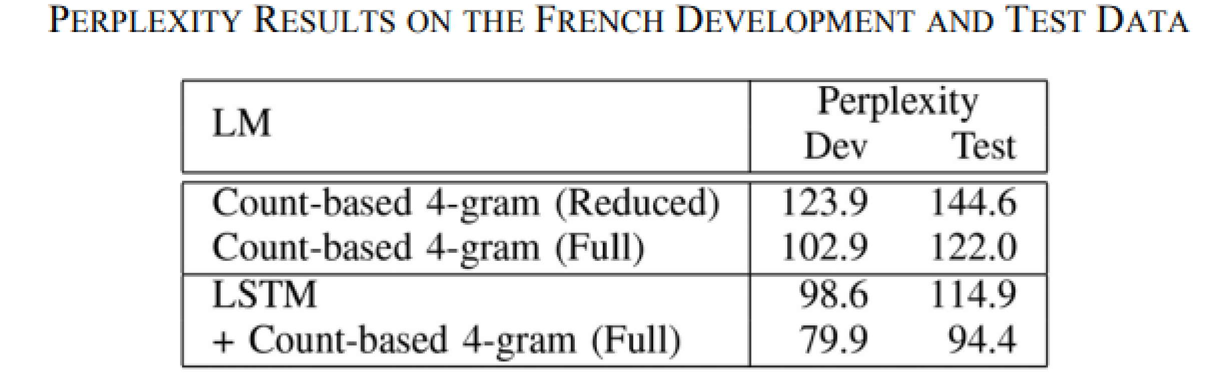


	n	c	h	m	direct	mix	train.	valid.	test.
MLP1	5		50	60	yes	no	182	284	268
MLP2	5		50	60	yes	yes		275	257
MLP3	5		0	60	yes	no	201	327	310
MLP4	5		0	60	yes	yes		286	272
MLP5	5		50	30	yes	no	209	296	279
MLP6	5		50	30	yes	yes		273	259
MLP7	3		50	30	yes	no	210	309	293
MLP8	3		50	30	yes	yes		284	270
MLP9	5		100	30	no	no	175	280	276
MLP10	5		100	30	no	yes		265	**252**
Del. Int.	3						31	352	336
Kneser-Ney back-off	3							334	323
Kneser-Ney back-off	4							332	321
Kneser-Ney back-off	5							332	321
class-based back-off	3	150						348	334
class-based back-off	3	200						354	340
class-based back-off	3	500						326	**312**
class-based back-off	3	1000						335	319
class-based back-off	3	2000						343	326
class-based back-off	4	500						327	312
class-based back-off	5	500						327	312

Bengio, Y., Ducharme, R., & Vincent, P. (2000). A neural probabilistic language model. *Advances in neural information processing systems*, *13*.

https://www.jmlr.org/papers/volume3/bengio03a/bengio03a.pdf

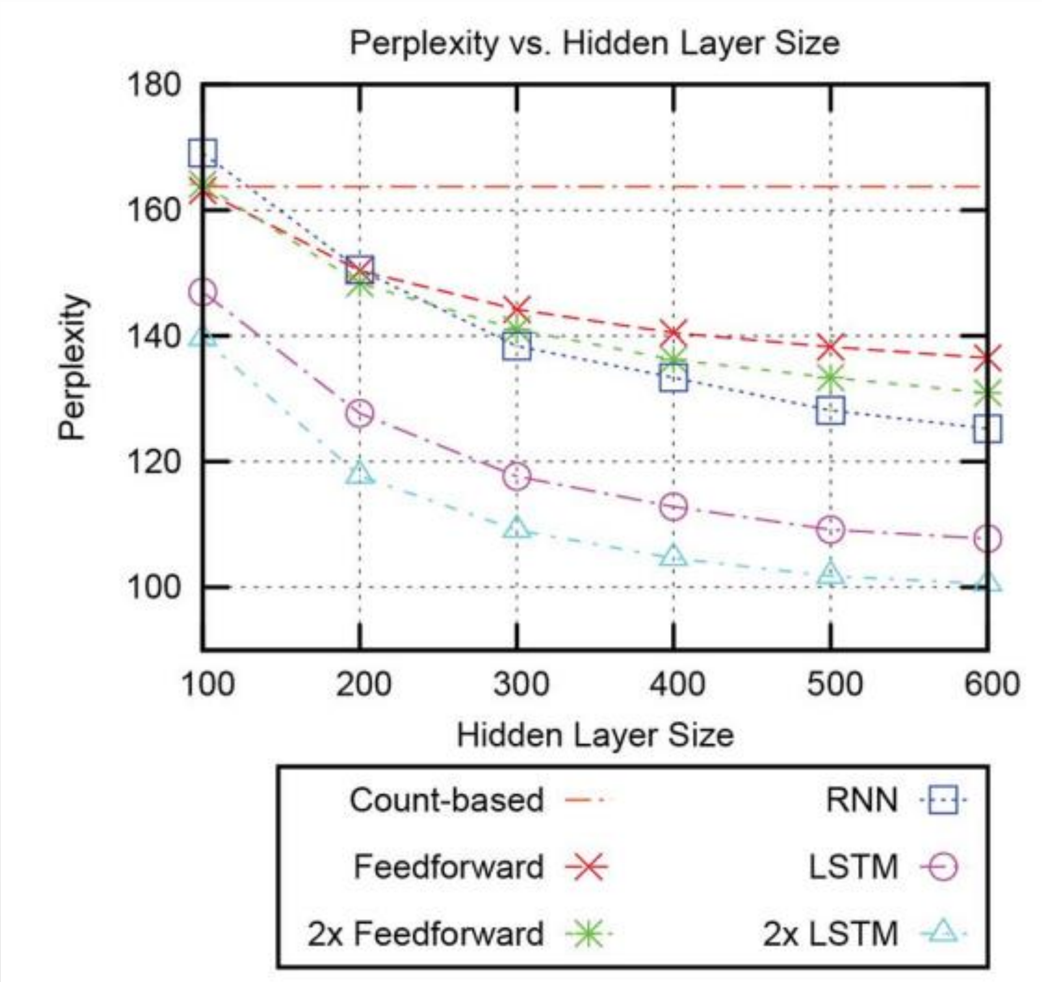
Neural Language Model (cont.)



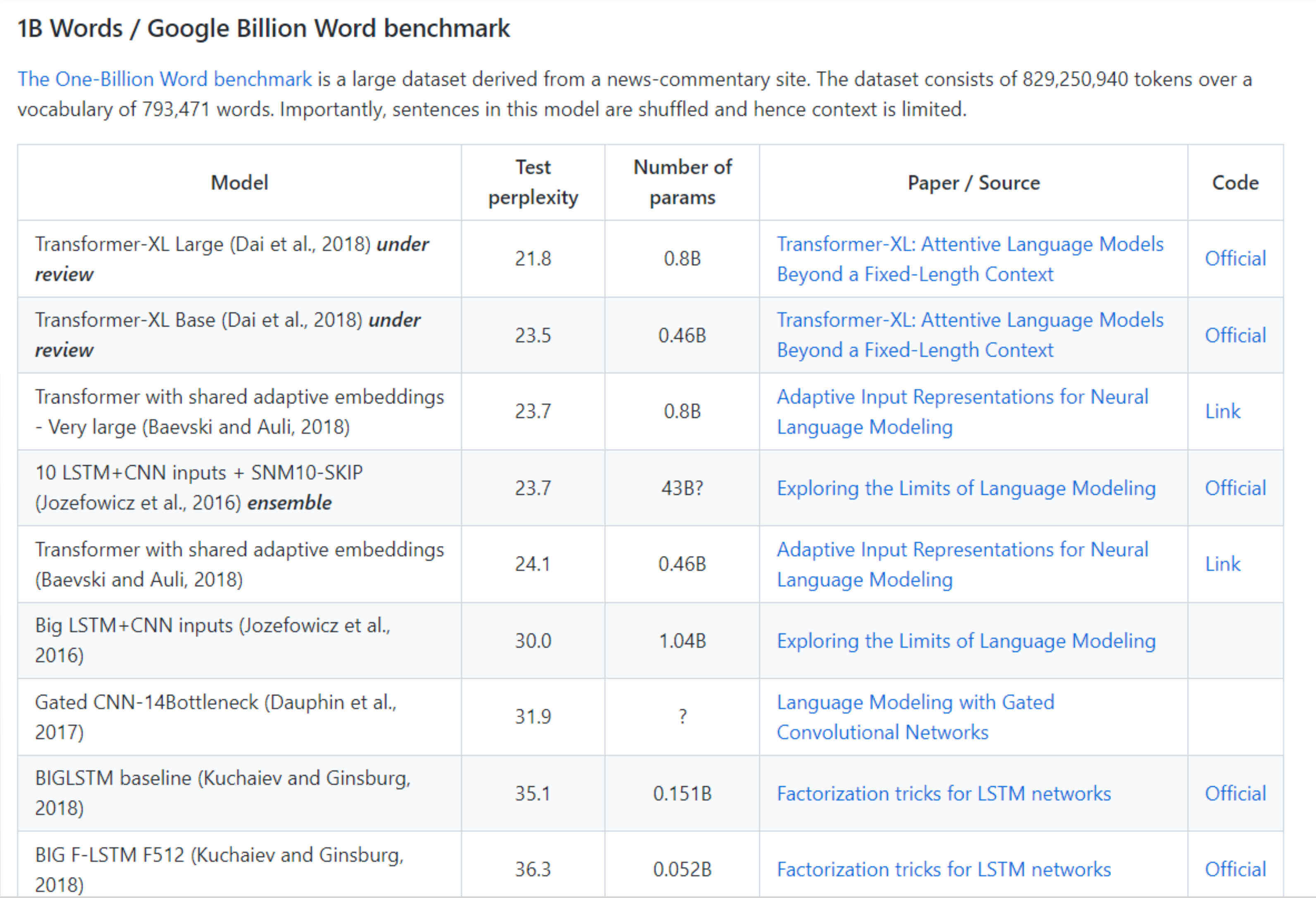
PERPLEXITY RESULTS ON THE FRENCH DEVELOPMENT AND TEST DATA

LM	Perplexity	
	Dev	Test
Count-based 4-gram (Reduced)	123.9	144.6
Count-based 4-gram (Full)	102.9	122.0
LSTM	98.6	114.9
+ Count-based 4-gram (Full)	79.9	94.4

Sundermeyer, M., Ney, H., & Schlüter, R. (2015). From feedforward to recurrent LSTM neural networks for language modeling. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, *23*(3), 517-529.



Neural Language Model (cont.)



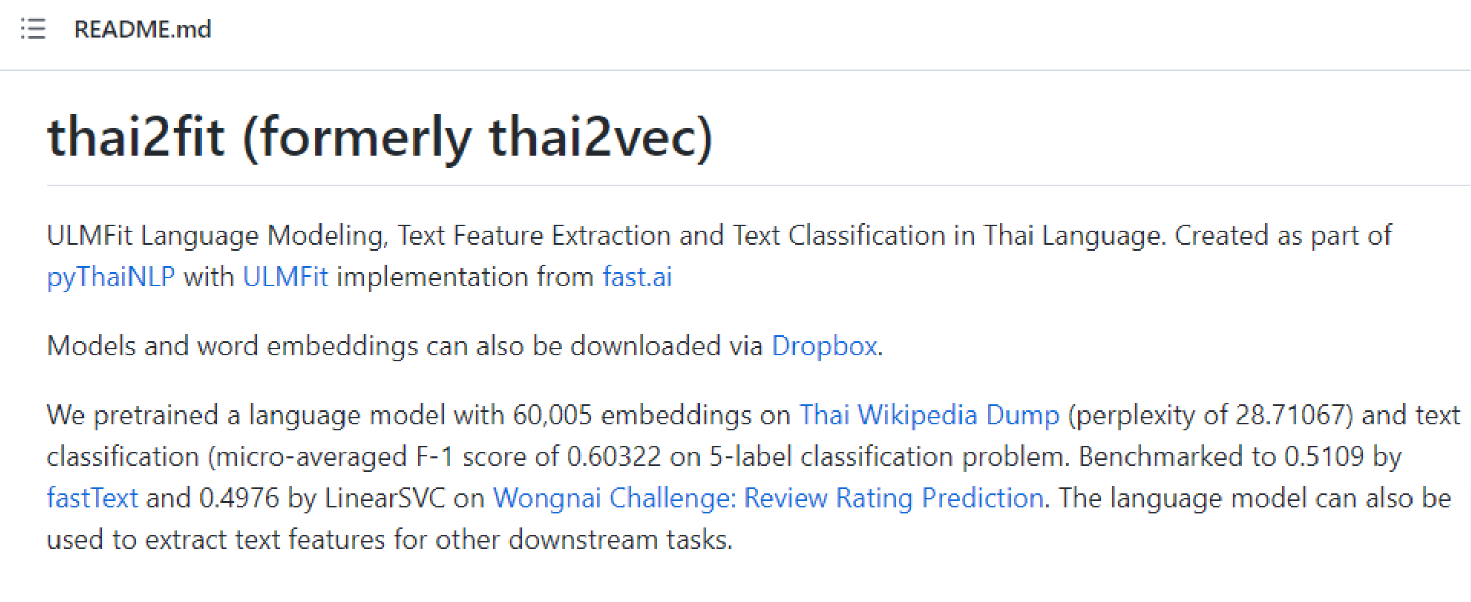
1B Words / Google Billion Word benchmark

The One-Billion Word benchmark is a large dataset derived from a news-commentary site. The dataset consists of 829,250,940 tokens over a vocabulary of 793,471 words. Importantly, sentences in this model are shuffled and hence context is limited.

Model	Test perplexity	Number of params	Paper / Source	Code
Transformer-XL Large (Dai et al., 2018) ***under review***	21.8	0.8B	Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context	Official
Transformer-XL Base (Dai et al., 2018) ***under review***	23.5	0.46B	Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context	Official
Transformer with shared adaptive embeddings - Very large (Baevski and Auli, 2018)	23.7	0.8B	Adaptive Input Representations for Neural Language Modeling	Link
10 LSTM+CNN inputs + SNM10-SKIP (Jozefowicz et al., 2016) ***ensemble***	23.7	43B?	Exploring the Limits of Language Modeling	Official
Transformer with shared adaptive embeddings (Baevski and Auli, 2018)	24.1	0.46B	Adaptive Input Representations for Neural Language Modeling	Link
Big LSTM+CNN inputs (Jozefowicz et al., 2016)	30.0	1.04B	Exploring the Limits of Language Modeling	
Gated CNN-14Bottleneck (Dauphin et al., 2017)	31.9	?	Language Modeling with Gated Convolutional Networks	
BIGLSTM baseline (Kuchaiev and Ginsburg, 2018)	35.1	0.151B	Factorization tricks for LSTM networks	Official
BIG F-LSTM F512 (Kuchaiev and Ginsburg, 2018)	36.3	0.052B	Factorization tricks for LSTM networks	Official

https://github.com/sebastianruder/NLP-progress/blob/master/english/language_modeling.md

Neural Language Model (cont.)



README.md

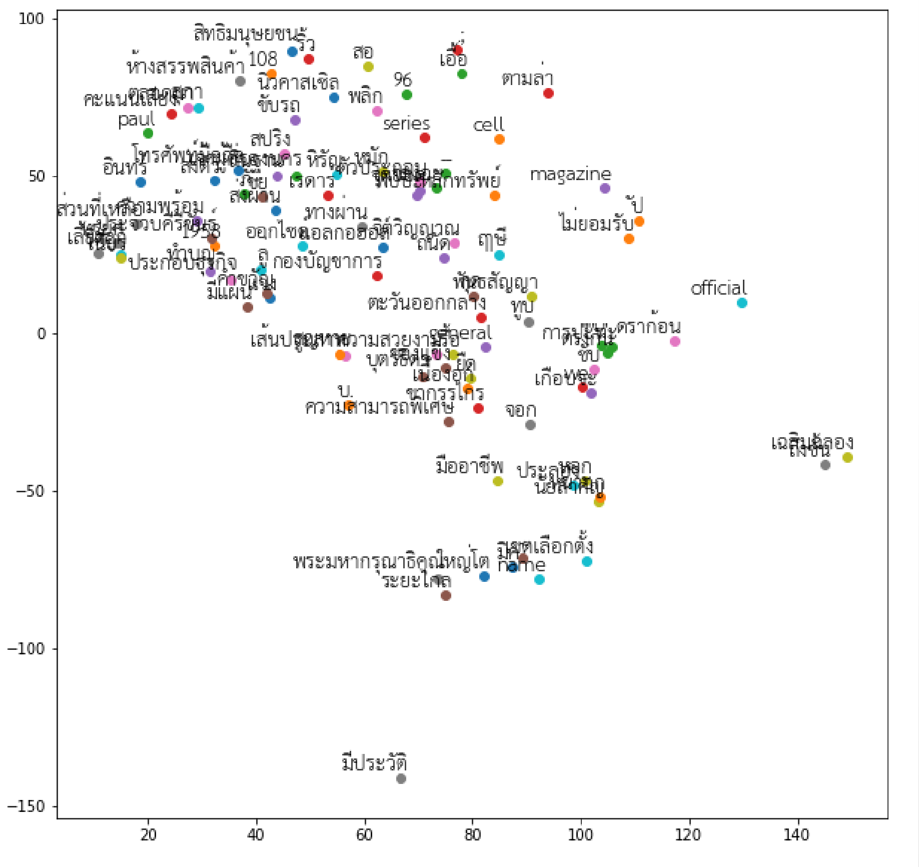
thai2fit (formerly thai2vec)

ULMFit Language Modeling, Text Feature Extraction and Text Classification in Thai Language. Created as part of pyThaiNLP with ULMFit implementation from fast.ai

Models and word embeddings can also be downloaded via Dropbox.

We pretrained a language model with 60,005 embeddings on Thai Wikipedia Dump (perplexity of 28.71067) and text classification (micro-averaged F-1 score of 0.60322 on 5-label classification problem. Benchmarked to 0.5109 by fastText and 0.4976 by LinearSVC on Wongnai Challenge: Review Rating Prediction. The language model can also be used to extract text features for other downstream tasks.

https://github.com/cstorm125/thai2fit



Neural Language Model (cont.)

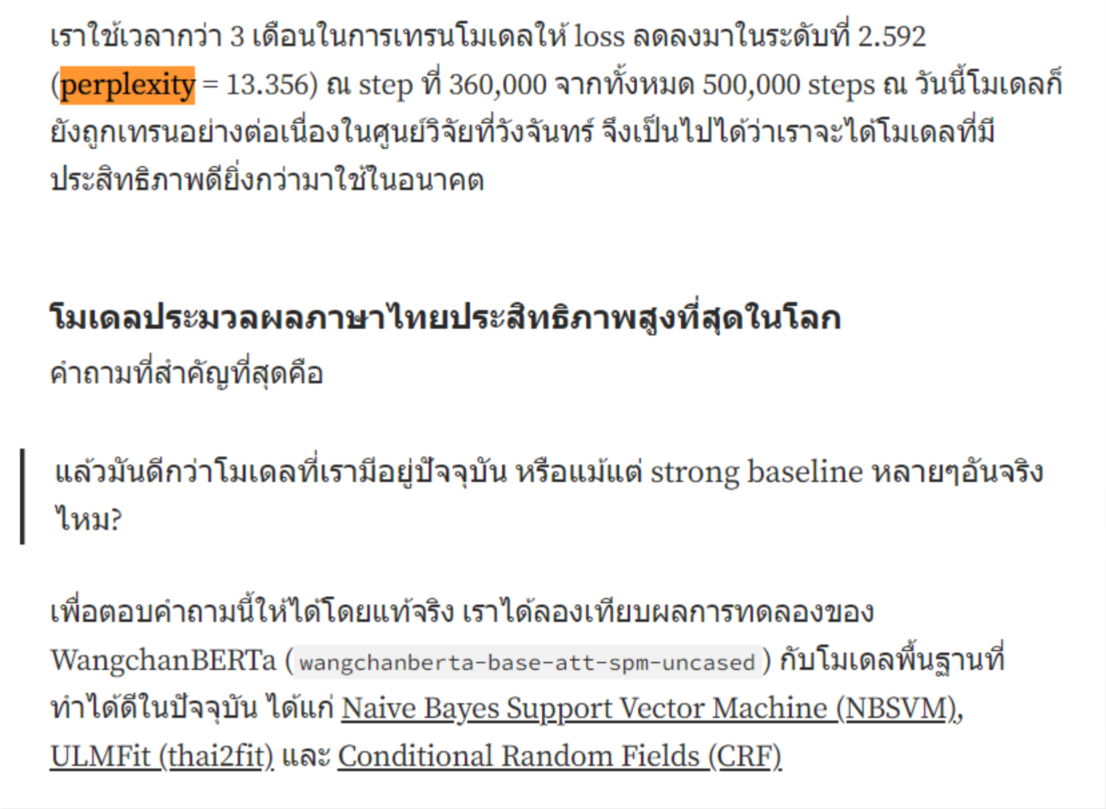
VISTEC-depa AI Research Institute of Thailand

Jan 24, 2021 · 5 min read

WangchanBERTa โมเดลประมวลผลภาษาไทยที่ใหญ่และก้าวหน้าที่สุดในขณะนี้

> *เปิดให้ทุกคนใช้ฟรีโดย AIResearch.in.th และ VISTEC ภายใต้สัญญาอนุญาต CC-BY-SA 4.0*

Image by Phannisa Nirattiwongsakorn



เราใช้เวลากว่า 3 เดือนในการเทรนโมเดลให้ loss ลดลงมาในระดับที่ 2.592 (perplexity = 13.356) ณ step ที่ 360,000 จากทั้งหมด 500,000 steps ณ วันนี้โมเดลก็ยังถูกเทรนอย่างต่อเนื่องในศูนย์วิจัยที่วังจันทร์ จึงเป็นไปได้ว่าเราจะได้โมเดลที่มีประสิทธิภาพดียิ่งกว่ามาใช้ในอนาคต

โมเดลประมวลผลภาษาไทยประสิทธิภาพสูงที่สุดในโลก

คำถามที่สำคัญที่สุดคือ

> แล้วมันดีกว่าโมเดลที่เรามีอยู่ปัจจุบัน หรือแม้แต่ strong baseline หลายๆอันจริงไหม?

เพื่อตอบคำถามนี้ให้ได้โดยแท้จริง เราได้ลองเทียบผลการทดลองของ WangchanBERTa (`wangchanberta-base-att-spm-uncased`) กับโมเดลพื้นฐานที่ทำได้ดีในปัจจุบัน ได้แก่ Naive Bayes Support Vector Machine (NBSVM), ULMFit (thai2fit) และ Conditional Random Fields (CRF)

Demo: LM

https://drive.google.com/file/d/1ArJvCcfP5bTqH6xzsbSHJaFwNPL3ouzz/view?usp=share_link