INTRODUCTION TO NLP

Session 5

David Buchaca Prats
2022

WHAT IS A LANGUAGE MODEL

- A Language Model is a probabilistic model that assigns probabilities to sequences of words. Therefore, we can use a language model to assign a numerical value (probability) to a given string (after tokenising).

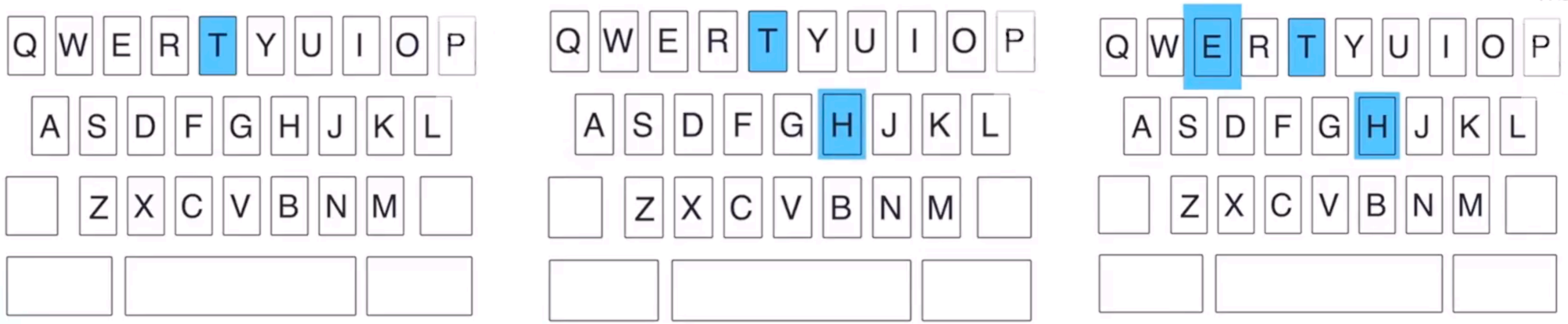
$$p(w_1, \ldots, w_n) = \prod_{i=1}^{n} p(w_i \mid w_1, \ldots, w_{n-1})$$

- Examples of problems that benefit from Language Models:
  - Spelling Correction.
  - Machine Translation.
  - Speech Recognition.
  - Handwriting recognition.
  - Word Suggestions in text editors.

A VERY IMPORTANT APPLICATION OF LANGUAGE MODELS

- Most tactile keyboards before the iPhone were bad.
- Users could make mistakes by simply hitting a region nearby a letter but outside the key that contains a letter.

- The iphone introduced dynamically adjusted hit regions in the keyboard.



CONSTRUCTING A LANGUAGE MODEL

- Let us consider a toy corpus from: https://en.wikipedia.org/wiki/This_Is_the_House_That_Jack_Built

  This is the house that Jack built
  This is the malt
  That lay in the house that Jack built
  This is the rat
  That ate the malt
  That lay in the house that Jack built
  This is the cat
  That killed the rat
  That ate the malt
  That lay in the house that Jack build

CONSTRUCTING A LANGUAGE MODEL

- We want to compute a bi-gram language model, which assumes $p(w_n \mid w_1, \ldots, w_{n-1}) \approx p(w_n \mid w_{n-1}) = p(w_n, w_{n-1}) / p(w_{n-1})$

This is the house that Jack built
This is the malt
That lay in the house that Jack built
This is the rat
That ate the malt
That lay in the house that Jack built
This is the cat
That killed the rat
That ate the malt
That lay in the house that Jack build

What could be a sensible way to estimate $p(w_i, w_j)$?

MARKOV ASSUMPTION

- N-gram language models make the Markov assumption, which we already saw in the hidden states of the HMM.
- Let us consider $x = (w_1, w_2, w_3, \ldots, w_N)$
- Using the Chain rule of probabilities:
  - $p(x) = p(w_1)p(w_2 \mid w1)p(w_3 \mid w_2, w_1)\ldots p(w_N \mid w_{N-1}, \ldots, w_2, w_1)$
- A k-Markov assumption assumes:
  - $p(w_n \mid w_1, \ldots, w_{n-1}) = p(w_n \mid w_{n-k+1}, \ldots, w_{n-1})$

EXAMPLES OF MARKOV ASSUMPTIONS

- A k-Markov assumption assumes:
  - $p(w_n \mid w_1, \ldots, w_{n-1}) = p(w_n \mid w_{n-k+1}, \ldots, w_{n-1})$
- Example: if k=2 and n=4
  - $p(w_4 \mid w_1, w_2, w_3) = p(w_4 \mid w_{n-k+1}, \ldots, w_{n-1}) = p(w_4 \mid w_{4-2+1}, \ldots, w_{4-1}) = p(w_4 \mid w_3)$
- Example: k=3 and n=4
  - $p(w_4 \mid w_1, w_2, w_3) = p(w_4 \mid w_{n-k+1}, \ldots, w_{n-1}) = p(w_4 \mid w_{4-3+1}, \ldots, w_{4-1}) = p(w_4 \mid w_2, w_3)$

A NOTE ON STORING COUNTS FOR AN N-GRAM LANGUAGE MODEL

- An expression of the form $p(w_n \mid w_{n-1}) = \frac{C(w_{n-1}w_n)}{C(w_{n-1})}$ requires us to store the counts of pairs of words and single words.
- There is an alternative:
  - $p(w_n \mid w_{n-1}) = \frac{C(w_{n-1}w_n)}{C(w_{n-1})} = \frac{C(w_{n-1}w_n)}{\sum_{\tilde{w}} C(w_{n-1}\tilde{w})}$
  - The expression that marginalises over $\tilde{w}$ has the advantage that needs less memory but the disadvantage that it requires summing over $\tilde{w}$ at runtime.

CONSTRUCTING A LANGUAGE MODEL

- We want to compute a bi-gram language model, which assumes $p(w_n \mid w_1, \ldots, w_{n-1}) \approx p(w_n \mid w_{n-1}) = p(w_n, w_{n-1}) / p(w_{n-1})$

This is the house that Jack built
This is the malt
That lay in the house that Jack built
This is the rat
That ate the malt
That lay in the house that Jack built
This is the cat
That killed the rat
That ate the malt
That lay in the house that Jack build

What could be a sensible way to estimate $p(w_i \mid w_j)$?

Count how many times (w_i, w_j) appears in the corpus and divide by the counts of (w_j)

CONSTRUCTING A LANGUAGE MODEL

- We want to compute a bi-gram language model, which assumes $p(w_n \mid w_1, \ldots, w_{n-1}) \approx p(w_n \mid w_{n-1}) = p(w_n, w_{n-1})/p(w_{n-1})$

This is the house that Jack built
This is the malt
That lay in the house that Jack built
This is the rat
That ate the malt
That lay in the house that Jack built
This is the cat
That killed the rat
That ate the malt
That lay in the house that Jack build

What could be a sensible way to estimate $p(w_i \mid w_j)$?

Count how many times (w_i, w_j) appears in the corpus and divide by the counts of (w_j)

$$p(\text{house} \mid \text{the}) = \frac{c(\text{the house})}{c(\text{the})} = ?$$

CONSTRUCTING A LANGUAGE MODEL

➤ We want to compute a bi-gram language model, which assumes $p(w_n | w_1, \ldots, w_{n-1}) \approx p(w_n | w_{n-1}) = p(w_n, w_{n-1}) / p(w_{n-1})$

This is the house that Jack built
This is the malt
That lay in the house that Jack built
This is the rat
That ate the malt
That lay in the house that Jack built
This is the cat
That killed the rat
That ate the malt
That lay in the house that Jack build

What could be a sensible way to estimate $p(w_i | w_j)$?

Count how many times (w_i, w_j) appears in the corpus and divide by the counts of (w_j)

$$p(\text{house} | \text{the}) = \frac{c(\text{the house})}{c(\text{the})} = ?$$

CONSTRUCTING A LANGUAGE MODEL

- We want to compute a bi-gram language model, which assumes $p(w_n \mid w_1, \ldots, w_{n-1}) \approx p(w_n \mid w_{n-1}) = p(w_n, w_{n-1})/p(w_{n-1})$

This is the house that Jack built
This is the malt
That lay in the house that Jack built
This is the rat
That ate the malt
That lay in the house that Jack built
This is the cat
That killed the rat
That ate the malt
That lay in the house that Jack build

What could be a sensible way to estimate $p(w_i \mid w_j)$?

Count how many times (w_i, w_j) appears in the corpus and divide by the counts of (w_j)

$$p(\text{house} \mid \text{the}) = \frac{c(\text{the house})}{c(\text{the})} = \frac{4}{10}$$

CONSTRUCTING A LANGUAGE MODEL

- We want to compute a 4-gram language model, which assumes
  $p(w_n \mid w_1, \ldots, w_{n-1}) \approx p(w_n \mid w_{n-2}, w_{n-1}) = p(w_{n-3}, w_{n-2}, w_{n-1}, w_n)/p(w_{n-3}, w_{n-2}, w_{n-1})$

This is the house that Jack built
This is the malt
That lay in the house that Jack built
This is the rat
That ate the malt
That lay in the house that Jack built
This is the cat
That killed the rat
That ate the malt
That lay in the house that Jack build

$$p(\text{house} \mid \text{this is the}) = \frac{c(\text{this is the house})}{c(\text{this is the})} = ?$$

CONSTRUCTING A LANGUAGE MODEL

- We want to compute a 4-gram language model, which assumes
  $p(w_n \mid w_1, \ldots, w_{n-1}) \approx p(w_n \mid w_{n-2}, w_{n-1}) = p(w_{n-3}, w_{n-2}, w_{n-1}, w_n) / p(w_{n-3}, w_{n-2}, w_{n-1})$

This is the house that Jack built
This is the malt
That lay in the house that Jack built
This is the rat
That ate the malt
That lay in the house that Jack built
This is the cat
That killed the rat
That ate the malt
That lay in the house that Jack build

$$p(\text{house} \mid \text{this is the}) = \frac{c(\text{this is the house})}{c(\text{this is the})} = ?$$

CONSTRUCTING A LANGUAGE MODEL

- We want to compute a 4-gram language model, which assumes $p(w_n \mid w_1, \ldots, w_{n-1}) \approx p(w_n \mid w_{n-2}, w_{n-1}) = p(w_{n-3}, w_{n-2}, w_{n-1}, w_n) / p(w_{n-3}, w_{n-2}, w_{n-1})$

This is the house that Jack built
This is the malt
That lay in the house that Jack built
This is the rat
That ate the malt
That lay in the house that Jack built
This is the cat
That killed the rat
That ate the malt
That lay in the house that Jack build

$$p(\text{house} \mid \text{this is the}) = \frac{c(\text{this is the house})}{c(\text{this is the})} = \frac{1}{4}$$

LANGUAGE MODELS WITH <S> AND </S>

- In order to deal with n-grams that affect the beginning or the end of a sentence the special words <s> and </s> are used to denote the start and stop of a sentence.
- Using this special words will affect the model probabilities.
- Consider the corpus:
  <s> This is the malt </s>
  <s> That lay in the house that Jack built </s>
- A 2-gram language model without <s> and </s> will assign to 'this is the house':

$$p(\text{this is the house}) = p(\text{this})p(\text{is} \mid \text{this})p(\text{the} \mid \text{is})p(\text{house} \mid \text{the}) = \frac{1}{12}\frac{1}{1}\frac{1}{1}\frac{1}{2} = \frac{1}{24}$$

- A 2-gram language model with <s> and </s> will assign to 'this is the house':

$$p(<s> \text{this is the house} </s>) = p(\text{this} \mid \text{start})p(\text{is} \mid \text{this})p(\text{the} \mid \text{is})p(\text{house} \mid \text{the})p(\text{end} \mid \text{house}) = ?$$

LANGUAGE MODELS WITH <S> AND </S>

- In order to deal with n-grams that affect the beginning or the end of a sentence the special words <s> and </s> are used to denote the start and stop of a sentence.
- Using this special words will affect the model probabilities.
- Consider the corpus:
  <s> This is the malt </s>
  <s> That lay in the house that Jack built </s>
- A 2-gram language model without <s> and </s> will assign to ‘this is the house’:

$$p(\text{this is the house}) = p(\text{this})p(\text{is}\,|\,\text{this})p(\text{the}\,|\,\text{is})p(\text{house}\,|\,\text{the}) = \frac{1}{12}\frac{1}{1}\frac{1}{1}\frac{1}{2} = \frac{1}{24}$$

- A 2-gram language model with <s> and </s> will assign to ‘this is the house’:

$$p(<\text{s}>\ \text{this is the house}\ <\text{/s}>\ ) = p(\text{this}\,|\,\text{start})p(\text{is}\,|\,\text{this})p(\text{the}\,|\,\text{is})p(\text{house}\,|\,\text{the})p(\text{end}\,|\,\text{house}) =$$

$$\frac{1}{2}\cdot\frac{1}{1}\cdot\frac{1}{1}\cdot\frac{1}{2}\cdot 0 = 0$$

DEALING WITH UNSEEN N-GRAMS: LAPLACE SMOOTHING

- We want to allow unseen grams to have non zero probability. We will pretend we saw each word one more time than we did.
- A common technique is called “Laplace Smoothing” or “add 1 smoothing”
  - Consist on using the following formulas:

$$p_{add_1}(w_n \mid w_{n-1}) = \frac{c(w_{n-1}w_n) + 1}{c(w_{n-1}) + V} \qquad p_{add_1}(w_i) = \frac{c(w_i) + 1}{c() + V} = \frac{c(w_i) + 1}{n_{tokens} + V}$$

- Consider the corpus:
  <s> This is the malt </s>
  <s> That lay in the house that Jack built </s>
- With V = 13 = #{ This, is, the, malt, That, lay, in, house, that, Jack, build, start, stop}
- $p(<\text{s}>\text{this is the house}</\text{s}>) = p(\text{this} \mid <\text{s}>)p(\text{is} \mid \text{this})p(\text{the} \mid \text{is})p(\text{house} \mid \text{the})p(</\text{s}> \mid \text{house}) =$
  ...

DEALING WITH UNSEEN N-GRAMS: LAPLACE SMOOTHING

- We want to allow unseen grams to have non zero probability. We will pretend we saw each word one more time than we did.
- A common technique is called “Laplace Smoothing” or “add 1 smoothing”
  - Consist on using the following formulas:

$$p_{add_1}(w_n | w_{n-1}) = \frac{c(w_{n-1}w_n) + 1}{c(w_{n-1}) + V} \qquad p_{add_1}(w_i) = \frac{c(w_i) + 1}{c() + V} = \frac{c(w_i) + 1}{n_{tokens} + V}$$

- Consider the corpus:
  <s> This is the malt </s>
  <s> That lay in the house that Jack built </s>
- With V = 13 = #{ This, is, the, malt, That, lay, in, house, that, Jack, build, start, stop}
- $p(<\text{s}>\text{this is the house} <\text{/s}>) = p(\text{this} | <\text{s}>)p(\text{is} | \text{this})p(\text{the} | \text{is})p(\text{house} | \text{the})p(<\text{/s}> | \text{house}) = \ldots$

$$\frac{1+1}{2+13} \cdot \frac{1+1}{1+13} \cdot \frac{1+1}{1+13} \cdot \frac{1+1}{2+13} \cdot \frac{0+1}{1+13}$$

DEALING WITH UNSEEN N-GRAMS: LIDSTONE SMOOTHING

- We want to allow unseen grams to have non zero probability. We will pretend we saw each word one more time than we did.
- A common technique is called "Lidstone Smoothing" .
  - Consist on using the following formulas:

$$p_{lindstone}(w_i | wj) = \frac{c(w_j w_i) + \epsilon}{c(w_j) + V \cdot \epsilon} \qquad p_{lindstone}(w_i) = \frac{c(w_i) + \epsilon}{c() + V \cdot \epsilon} = \frac{c(w_i) + \epsilon}{n_{tokens} + V \cdot \epsilon}$$

- Consider the corpus:
  <s> This is the malt </s>
  <s> That lay in the house that Jack built </s>
- With V = 13 = #{ This, is, the, malt, That, lay, in, house, that, Jack, build, start, stop}
- $p(< \text{s} > \text{this is the house} < /\text{s} >) = p(\text{this} | < \text{s} >)p(\text{is} | \text{this})p(\text{the} | \text{is})p(\text{house} | \text{the})p(< /\text{s} > | \text{house}) = \ldots$

DEALING WITH UNSEEN N-GRAMS: LIDSTONE SMOOTHING

- We want to allow unseen grams to have non zero probability. We will pretend we saw each word one more time than we did.
- A common technique is called "Lidstone Smoothing" or add-k smoothing
  - Consist on using the following formulas:

$$p_{lindstone}(w_i | wj) = \frac{c(w_j w_i) + k}{c(w_j) + V \cdot k} \qquad p_{lindstone}(w_i) = \frac{c(w_i) + k}{c() + V \cdot k} = \frac{c(w_i) + k}{n_{tokens} + V \cdot k}$$

- Consider the corpus:
  <s> This is the malt </s>
  <s> That lay in the house that Jack built </s>
- With V = 13 = #{ This, is, the, malt, That, lay, in, house, that, Jack, build, start, stop}
- $p(< \text{s} > \text{this is the house} < /\text{s} >) = p(\text{this} | < \text{s} >)p(\text{is} | \text{this})p(\text{the} | \text{is})p(\text{house} | \text{the})p(< /\text{s} > | \text{house}) = \ldots$

$$\frac{1+3}{2+13\cdot 3} \cdot \frac{1+3}{1+13\cdot 3} \cdot \frac{1+3}{1+13\cdot 3} \cdot \frac{1+3}{2+13\cdot 3} \cdot \frac{0+3}{1+13\cdot 3} \text{ if we use } k = 3$$

DEALING WITH UNSEEN N-GRAMS: LAPLACE AND LIDSTONE SMOOTHING

- Note that if we add a 1 to each of the counts of all the n-grams we would get:

$$p_{add_1}(w_n \mid w_{n-1}) = \frac{c(w_{n-1}w_n) + 1}{\sum_w (c(w_{n-1}w) + 1)} = \frac{c(w_{n-1}w_n) + 1}{\sum_w c(w_{n-1}w) + V} = \frac{c(w_{n-1}w_n) + 1}{c(w_{n-1}) + V}$$

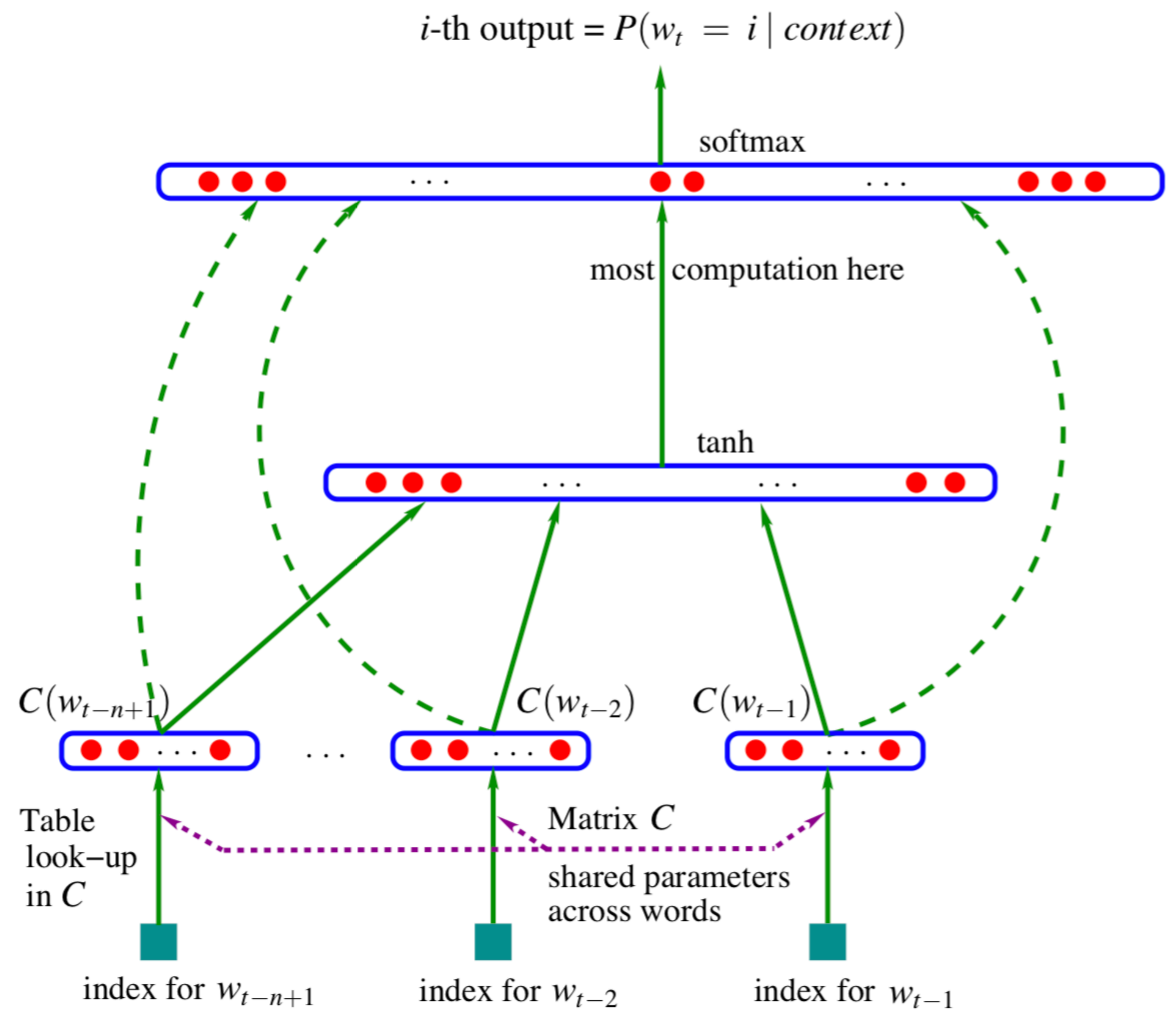
- In a similar way, if we add k to each of the n-gram counts we would get:

$$p_{add_k}(w_n \mid w_{n-1}) = \frac{c(w_{n-1}w_n) + k}{\sum_w (c(w_{n-1}w) + k)} = \frac{c(w_{n-1}w_n) + k}{\sum_w c(w_{n-1}w) + k \cdot V} = \frac{c(w_{n-1}w_n) + k}{c(w_{n-1}) + k \cdot V}$$

Add-k will make the probabilities smoother.

PROBLEMS WITH N-GRAM LANGUAGE MODELS

- They suffer from scalability issues:
  - Storing all counts for n-grams might require a lot of memory if N is big
  - 140 GB of ram for https://kheafield.com/papers/edinburgh/estimate_paper.pdf
- Neural Networks offer an alternative where RAM requirements scale with the number of words.



EXERCISE

- Given the corpus
  <s> I am Sam </s>
  <s> Sam I am </s>
  <s> I do not like green eggs and ham </s>

- Compute:
  P(I | <s>) =
  P(Sam | <s>) =
  P(am | I) =
  P(</s> | Sam) =
  P(Sam | am) =
  P(do| I) =

EXAMPLE

- Given the corpus
  <s> I am Sam </s>
  <s> Sam I am </s>
  <s> I do not like green eggs and ham </s>

- Compute:

- P(I | <s>) = 2/3
  P(Sam | <s>) = 1/3
  P(am | I) = 2/3
  P(</s> | Sam) = 1/2
  P(Sam | am) = 1/2
  P(do| I) = 1/3

NUMERICALL STABILITY WITH N-GRAM MODELS

- When computing the probability for a long sentence you end up multiplying a lot of probabilities over the n-grams. Each product of two small numbers becomes an even smaller number.

$$p(<\text{s}>\text{this is the house}</\text{s}>) = p(\text{this}\,|\,\text{start})p(\text{is}\,|\,\text{this})p(\text{the}\,|\,\text{is})p(\text{house}\,|\,\text{the})p(\text{end}\,|\,\text{house})$$

- This can yield to underflow:
  - 0.0001 * 0.0003 * 0.0003 * 0.0003 * 0.0005 = 1.349 e-18

COMPUTING IN LOG DOMAIN

- Computing in log domain allows us to deal with products of probabilities in a smart approach that can avoid numerical problems.
- If a number a is positive then $a = \exp(\log(a)) = \log(\exp(a))$
- Therefore we can compute:

$$p_1 \cdot p_2 \cdot p_3 = \exp(\log(p_1 \cdot p_2 \cdot p_3)) = \exp\left(\log(p_1) + \log(p_2) + \log(p_3)\right)$$

- Yeah but … if the previous equality is an equality why do we care??
  - Because summing $\log(p_i)$ will not result to numerical underflow but multiplying the p_i might.

EVALUATING LANGUAGE MODELS

- One of the most common metrics to evaluate a language model is perplexity.
- The perplexity of a language model on a test set is defined as the inverse probability of the test set, normalized by the number of words.
- Note that the higher the probability the lower the perplexity (thus we want low perplexity)
- For a test set $X_{te} = w_1 \dots w_N$ the perplexity of a model h on X_{te} is defined as:

$$PP(h; X_{te}) = P_h(w_1 \dots w_N)^{-\frac{1}{N}} = \sqrt[N]{\frac{1}{P_h(w_1 \dots w_N)}}$$

- If we use the chain rule to compute $P_h(w_1 \dots w_N)$ we have

$$PP(h; X_{te}) = \sqrt[N]{\frac{1}{P_h(w_1 \dots w_N)}} = \sqrt[N]{\frac{1}{\prod_{n=1}^{N} P_h(w_i \mid w_1 \dots w_{i-1})}} = \sqrt[N]{\prod_{n=1}^{N} \frac{1}{P_h(w_i \mid w_1 \dots w_{i-1})}}$$

EVALUATING LANGUAGE MODELS

- Note that in the definition of perplexity we used $X_{te} = w_1 \dots w_N$
- Here we mean the concatenation of all the lines in the test set. Since the test set will contain many sentence boundaries we need to include them in the probability computation.
- Given the test set
  <s> I am Sam </s>
  <s> Sam I am </s>
  <s> I do not like green eggs and ham </s>
- We would create "a single line test set":
- <s> I am Sam </s> <s> Sam I am </s><s> I do not like green eggs and ham </s>

EXAMPLE OF LANGUAGE MODEL USAGE

- Base algorithm for a simple spellchecker: Let x be a sentence and V the vocabulary.

```
For $w$ in $x$:
 If $w$ not in $V$:
   Find the closest words to w (candidate search)
   Evaluate each of the candidate words (candidate evaluation)
   Return the most probable candidate
```