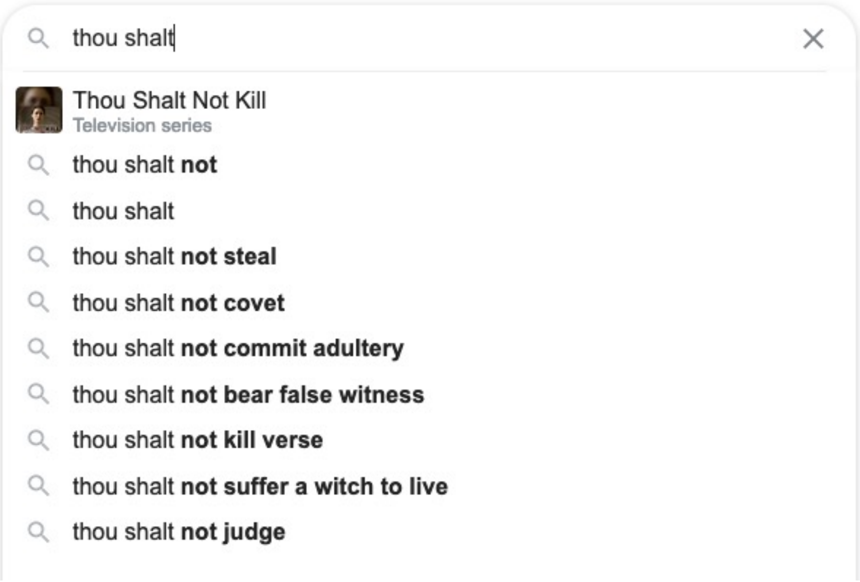
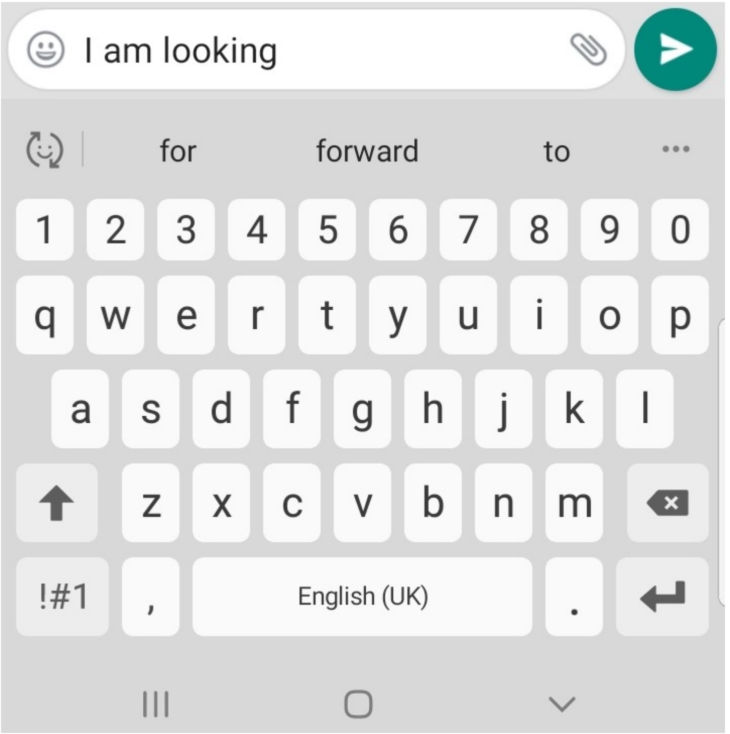
# Lecture 05:
# Language modelling

# Overview

- Language modelling
- Probabilistic Language modelling with N-gram
- Neural language model
- Evaluating language model

# Next word prediction

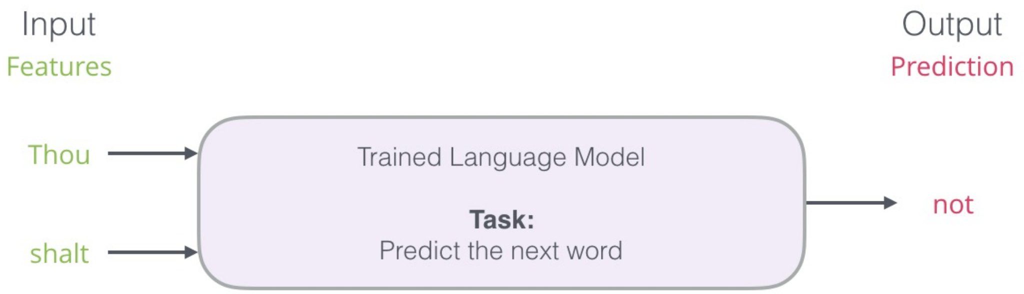


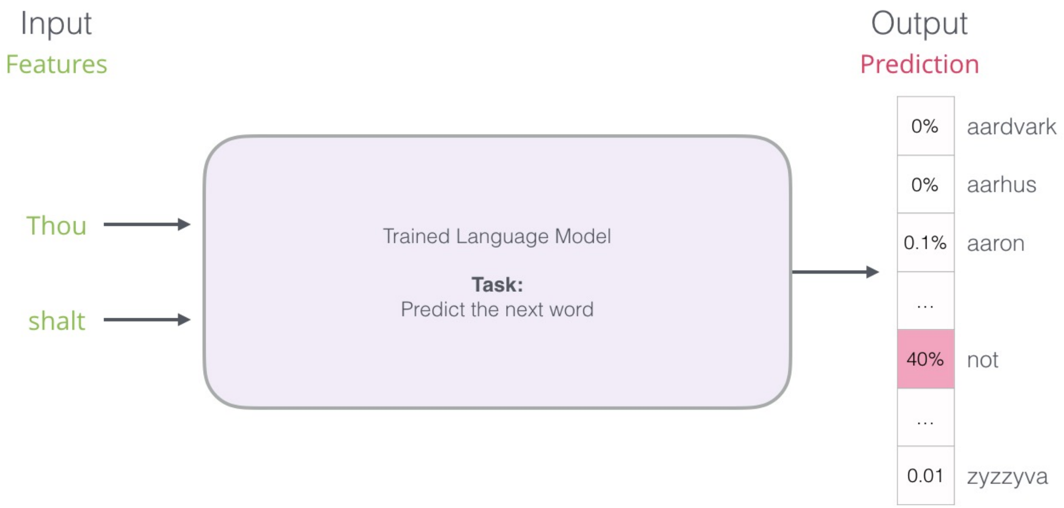



# Next word prediction

input/feature #1 input/feature #2 output/label

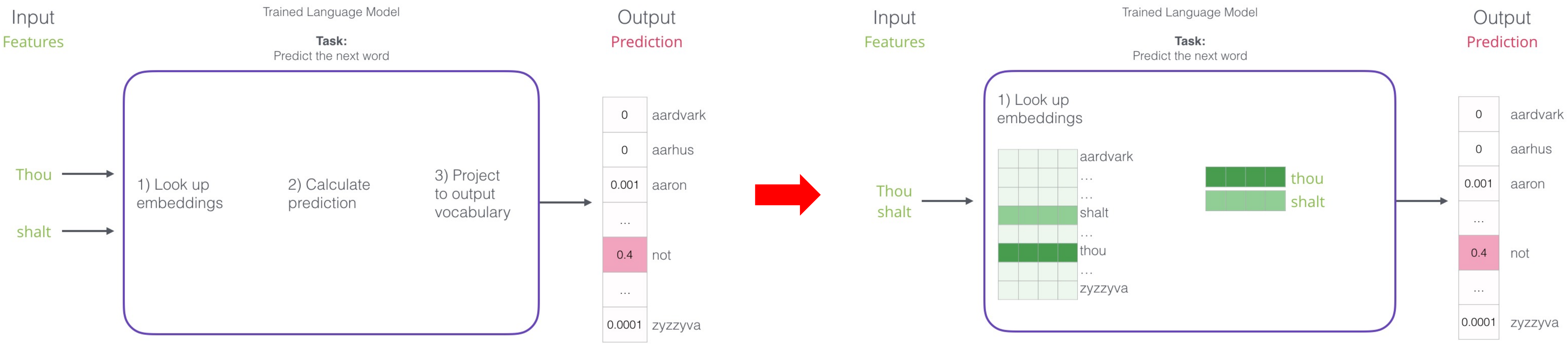
Thou shalt ____



# Language modeling

## Three main steps in a neural language models



# Language modeling

- Language modelling is the task of predicting the next word given an input sequence of words over a vocabulary *V.*
  - a system that assigns probabilities to a piece of text from a vocabulary *V.*

- Given a sequence of words $w^{(1)}, w^{(2)}, \ldots, w^{(t)}$, a language model compute the probability distribution of the next word $w^{(t+1)}$

$$P\left(w^{(t+1)} \middle| w^{(t)}, \ldots, w^{(1)}\right), \qquad \forall w \in V = \{w^1, \ldots, w^{|V|}\}$$

- Model that assigns a prbobality to the sequence of words

$$P(w^{(1)}, w^{(2)}, \ldots, w^{(t)})$$

# Language modelling frameworks

Currently there are two general techniques of training a language model thes can be classed as

- Traditional or probabilistic language model
  - N-gram model
    - Chain rule of probability
    - Markov assumption
- Neural language model
  - Based on neural networks (RNN)
  - LSTM, GRU,
    - Bidirectional RNN
    - Seq2seq (Encoder-decoder)

# Language model – n-grams

- N-gram language models are based on probabilities of chunks of word.

*The student open their*

- Definition: An n-gram is a chunk of n consecutive words.
  - unigram: unit of single word – “the”, “student”, “opened”, “their”
  - bigrams: unit of double words – “the student”, “student opened”, “opened their”
  - trigram: unit of triple words – “the student opened”, “student opened their”
  - 4-gram: unit of 4 words – “the student opened their”

- The main idea behind *n-gram* models is to collect statistics about the frequency of different *n-grams* and use this to predict the next word.

# N-gram language models

- Given a sequence of words $w^1, w^2, \dots, w^T$, the probability of this sequence occuring according to the language model is obtained using the chain rule:

$$P(w^1, w^2, \dots, w^T) = P(w^1) \text{ x } P(w^2|w^1) \text{ x } \dots \text{ x } P(w^T|w^{T-1}, \dots, w^1)$$

$$= \prod_{t=1}^{T} P(w^t|w^{t-1}, \dots, w^1)$$

# n-gram Language Models

## The chain rule

$$P(w_1, w_2, \ldots, w_{T-1}, w_T) = \prod_{t=1}^{T} P(w_t | w_{t-1}, w_{t-2}, \ldots, w_1)$$

| | | | | | | |
|---|---|---|---|---|---|---|
| **the** | cat | sat | on | the | mat | $P(w_1)$ |
| the | **cat** | sat | on | the | mat | $P(w_2 \| w_1)$ |
| the | cat | **sat** | on | the | mat | $P(w_3 \| w_2, w_1)$ |
| the | cat | sat | **on** | the | mat | $P(w_4 \| w_3, w_2, w_1)$ |
| the | cat | sat | on | **the** | mat | $P(w_5 \| w_4, w_3, w_2, w_1)$ |
| the | cat | sat | on | the | **mat** | $P(w_6 \| w_5, w_4, w_3, w_2, w_1)$ |

Slide credit: poitr Mirowski

# n-gram language model

*Markov assumptions:*

- $w^{(t+1)}$ depends only on the preceding *n-1* words.

$$P(w^{t+1}|w^{t},\dots,w^{1}) \approx P(w^{t+1}|w^{t},\dots,w^{t-n+2})$$

- By conditional probability theory we get

Prob of n-gram

$$P(w^{t+1}|w^{t},\dots,w^{t-n+2}) = \frac{\boxed{P(w^{t+1},\dots,w^{t-n+2})}}{\boxed{P(w^{t},\dots,w^{t-n+2})}}$$

Prob of (n-1)-gram

# n-gram Language Models

Markov assumption on n-gram

$$P(w_1, w_2, \ldots, w_{T-1}, w_T) \approx \prod_{t=1}^{T} P(w_t | w_{t-1}, \ldots, w_{t-n+1})$$

| | | | | | | |
|---|---|---|---|---|---|---|
| **the** | cat | sat | on | the | mat | $P(w_1)$ |
| the | **cat** | sat | on | the | mat | $P(w_2 \| w_1)$ |
| the | cat | **sat** | on | the | mat | $P(w_3 \| w_2, w_1)$ |
| the | cat | sat | **on** | the | mat | $P(w_4 \| w_3, w_2)$ |
| the | cat | sat | on | **the** | mat | $P(w_5 \| w_4, w_3)$ |
| the | cat | sat | on | the | **mat** | $P(w_6 \| w_5, w_4)$ |

Slide credit: poitr Mirowski

# n-gram Language Models

How do we get all these probabilities?

- The n-gram and *(n-1)-gram* probabilities are estimated by counting the frequency of occurance in the vocabulary:

$$\approx \frac{count(w^{t+1}, w^{t}, \dots, w^{t-n+2})}{count(w^{t}, \dots, w^{t-n+2})}$$

- Sequence frequencey of occurrence is a reasonable estimate of the probability

# n-gram Language Models: Example

Assume we are learning a ***trigram*** language model.

~~*By 10am this morning, Joe was out*~~ *waiting for the*_____

**discard**

**Condition on the previous 3 wouds**

<u>Example: given we have a coupus</u>

- "*waiting for the*" occurred 1000 times
- "*waiting for the bus*" occurred *350* times
- "*waiting for the car*" occurred 100 times
- Then we get that
  - P(bus| *waiting for the*) = 0.35
  - P(car| *waiting for the*) = 0.10

$$P(\boldsymbol{w}|waiting\ for\ the) = \frac{count(waiting\ for\ the\ \boldsymbol{w})}{count(waiting\ for\ the)}$$

# From n-gram probabilities to language model

- A language $L \subseteq V^*$ is a (possibly infinite) set of strings over a (finite) vocabulary $V$.

- $P(w^t|w^{t-1})$ defines a distribution over all *2-grams* in $V$:

$$\forall w \in V : \sum_{w' \in V} P(w^t = w'|w^{t-1} = w) = 1$$

- By multiplying this distribution $N$ times we get one distribution over all strings of the same length $N$ $(VN)$:
  - Probability of one *N*-word string: $P(w_1 \ldots w_N) = \prod_{i=1}^{N} P\left(w^{(i)} = w_i \middle| w^{(i-1)} = w_{i-1}\right)$
  - Probability of all *N*-word string: $P(V^N) = \sum_{w,w' \in V} \prod_{i=1}^{N} P\left(w^{(i)} = w \middle| w^{(i-1)} = w'\right)$

# From n-gram probabilities to language model

A language model $P(L) = P(V^*)$ should define one distribution $P(V^*)$ that sums to one over *all* strings in *L* ⊆ *V**, *regardless of their length:*

$$P(L) = P(V^1) + P(V^2) + P(V^3) + \ldots + P(V^n)$$

**Solution:**

Add ***end-of-Sentence (EOS) or beginning-of-sentence (BOS)*** token to vocabulary *V,* assumptions;

- Each string ends with EOS (or BOS if focus is on start of string)
- EOS can only appear at the end of a string (or BOS at start of string)

# From n-gram probabilities to language model

In a trigram model

$$P(w^1 w^2 w^3) = P(w^1)P(w^2|w^1)P(w^3|w^2, w^1)$$

- The only trigram is $P(w^3|w^2, w^1)$
- $P(w^1)$ and $P(w^2|w^1)$ are not trigrams

- Add *n-1* BOS symbols to the sentence for an n-gram model

$$BOS_1 BOS_2 w^1 w^2, \ldots, w^n$$

# From n-gram probabilities to language model

- A language model can be regarded as a *stochastic process:*
  - At each time step, randomly pick one token.
  - Stop when the word you picked is a special *EOS* token.
- Add an *EOS* to all sentences in your corpus, thus our vocabulary is now defined as

$$V^{EOS} = V \cup \{EOS\} \; or$$
$$V^{BOS} = V \cup \{BOS\}$$

- With the new vocabulary we can get a single distribution over strings of any length because $P(EOS| \ \dots)$ will have high enough that we are always guaranteed to stop after generating a finite number of words.

# Summary: training bi-gram model

- Replace all words not in training vocabulary by the unknown token *UNK*
- Enclose each sentence by special start and stop symbols

$$< s > Alice\ was\ beginning\ to\ get\ very\ tired\ \ldots . </s >$$

- Define new vocabulary $V' = \{V, UNK, < s >, </s >\}$
- Count the frequency of each bigram

$$C(< s >, Alice) = 1, C(Alice, was) = 1, \ldots$$

- and normalize these frequencies to get probabilities

$$P(was|Alice) = \sum_{w_i \in V'} \frac{C(Alice\ was)}{C(Alice\ w_i)}$$

# Evaluating Language model

There are two ways to evaluate langauge models:

- Intrinsic evaluation measures how ell the model captures what it is suppose to capture (e.g. probabilities)
- Extrinsic (task-based) evalaution measures the language model improves the performance of the particular task.

- Both cases requires an evaltion metric that allows us to measure and compare the performan.ce of different models

# Intrinsic evaluation

Intrinsic evaluation follows the normal procedure for every other machine learning model

- Define an evaluation metric (scoring function).
- Train the model on a training set
- Test the model on an unseen test set
- Compare language models by their scores.

# Intrinsic evaluation - perplexity

Perplexity is the probability of the test set normalized by the number of words:

$$PP(w_1 \ldots w_N) =_{def} \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_{i-1}, \ldots, w_{i-n+1})}}$$

with

$$PP(w_1 \ldots w_N) =_{def} \exp\left(-\frac{1}{N}\sum_{i=1}^{N} \log P(w_i|w_{i-1}, \ldots, w_{i-n+1}\right)$$

**Practical issues**

- Since language model probabilities are very small, multiplying them together often yields to arithmetic underflow.
- It is often better to use logarithms scale instead, so replace

# Intrinsic evaluation- perplexity

- **Perplexity:** most common intrinsic metric
  - based on Shannon principle
  - Perplexity is a bad approximation
    - Only use if the test data is very similar to the training set
    - Generally useful in pilot experiments

- A better model
  - Is one that assigns the higher probability to the word that occur

# Extrinsic evaluation

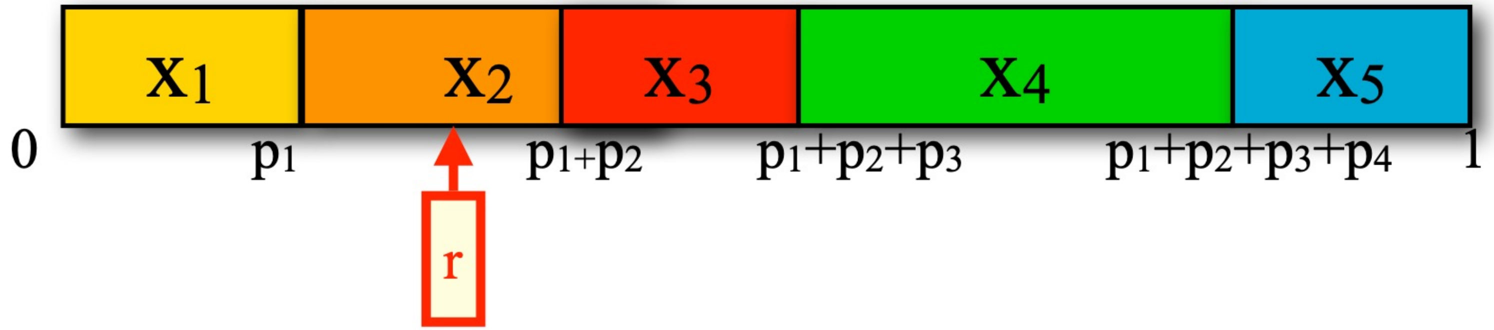
Best evaluation from comparing language model A and B

- Put A and B in a task
  - Spelling corrector, machine translator system, speech recognizer
- Run the task, get an accuracy when A is in the system
- Run the task again with B in the model, get the accuracy
  - How many misspelled words corrected properly
  - How many words were translated correctly
- Compare accuracy for A and B

# Generating text with language models

Given an n-gram language model with probability distribution $P(X|Y = y)$

- Let $X = \{x_1, \ldots, x_N\}$ be $N$ possible outcomes and $P(X = x_i\ |Y = y) = p_i$
- Divide the interval [0, 1] into $N$ intervals according to the probabilities of the outcomes
- Generate a random number $r \in [0, 1]$
- Return the $x_i$ whose interval the number $r$ lies within



# Generating text with n-gram models

Text generation with a trigram model trained with 1.7 million words corpus (Reuters)

*today the _______*

get probability distribution

| | |
|---|---|
| company | 0.153 |
| bank | 0.153 |
| price | 0.077 |
| italian | 0.039 |
| emirate | 0.039 |
| … | |

**Sparsity problem**: not much granularity in the probability distribution

*today the _______*

condition on this

get probability distribution

| | |
|---|---|
| company | 0.153 |
| bank | 0.153 |
| price | 0.077 |
| italian | 0.039 |
| emirate | 0.039 |
| … | |

sample

*today the price _______*

condition on this

get probability distribution

| | |
|---|---|
| of | 0.308 |
| for | 0.050 |
| it | 0.046 |
| to | 0.046 |
| is | 0.031 |
| … | |

sample

*today the price of _______*

condition on this

get probability distribution

| | |
|---|---|
| the | 0.072 |
| 18 | 0.043 |
| oil | 0.043 |
| its | 0.036 |
| gold | 0.018 |
| … | |

sample

*today the price of gold _______*

*today the price of gold per ton , while production of shoe lasts and shoe industry , the bank intervened just after it considered and rejected an imf demand to rebuild depleted european stocks , sept 30 end primary 76 cts a share .*

Surprisingly grammatical!

Credit: Chris Manning

# Limitations – number of possible parameters

Estimating the number of parameter per n-gram language model.

Given a vocabulary V of $|V|$ unique tokens, where $|V|$ = $10^4$

- **Unigram** model: $|V|$ parameters ⬄ $10^4$ parameters
  - One distribution $P(w^{(i)})$ with /V/ outcomes, each $w \in V$ is one outcome
- **Bigram** model: $|V|^2$ parameters ⬄ $10^8$ parameters
  - $|V|$ distribution $P\left(w^{(i)} \middle| w^{(i-1)}\right)$, one distribution for each $w \in V$ with |V| outcome each [each $w \in V$ is one outcome]
- **Trigram** model: $|V|$ parameters ⬄ $10^{12}$ parameters
  - $|V|$ distribution $P\left(w^{(i)} \middle| w^{(i-1)}, w^{(i-2)}\right)$, one distributionper bigram w'w' with |V| outcome each [each $w \in V$ is one outcome]

# Limitations – sparsity problem

**Sparsity Problem 1:**

- What if *"waiting for the w"* never occurred in the corpus?
- Then *w* has probability 0

**Solution - smoothing:**
Add small $\delta$ to the count for every $w \in V$.

$$P(\boldsymbol{w}|waiting\ for\ the) = \frac{\boxed{count(waiting\ for\ the\ \boldsymbol{w})}}{\boxed{count(waiting\ for\ the)}}$$

**Sparsity Problem 2:**

- What if *"waiting for the"* never occurred in the corpus?
- Then we cannot calculate probabiliity of any *w.*

**Solution - backoff:**
condition on the (n-1)-gram, i.e. *"waiting for"*

Larger *n* makes sparsity problem worse. Typically *n* should be less than or equal to 5

# Limitations – storage problems

**Storage Problem:**
The need to store count for all n-grams you saw in the corpus.

$$P(\boldsymbol{w}|\, waiting\ for\ the) = \frac{\boxed{count(waiting\ for\ the\ \boldsymbol{w})}}{count(waiting\ for\ the)}$$

Increasing *n* or increasing corpus size <=> increases model size