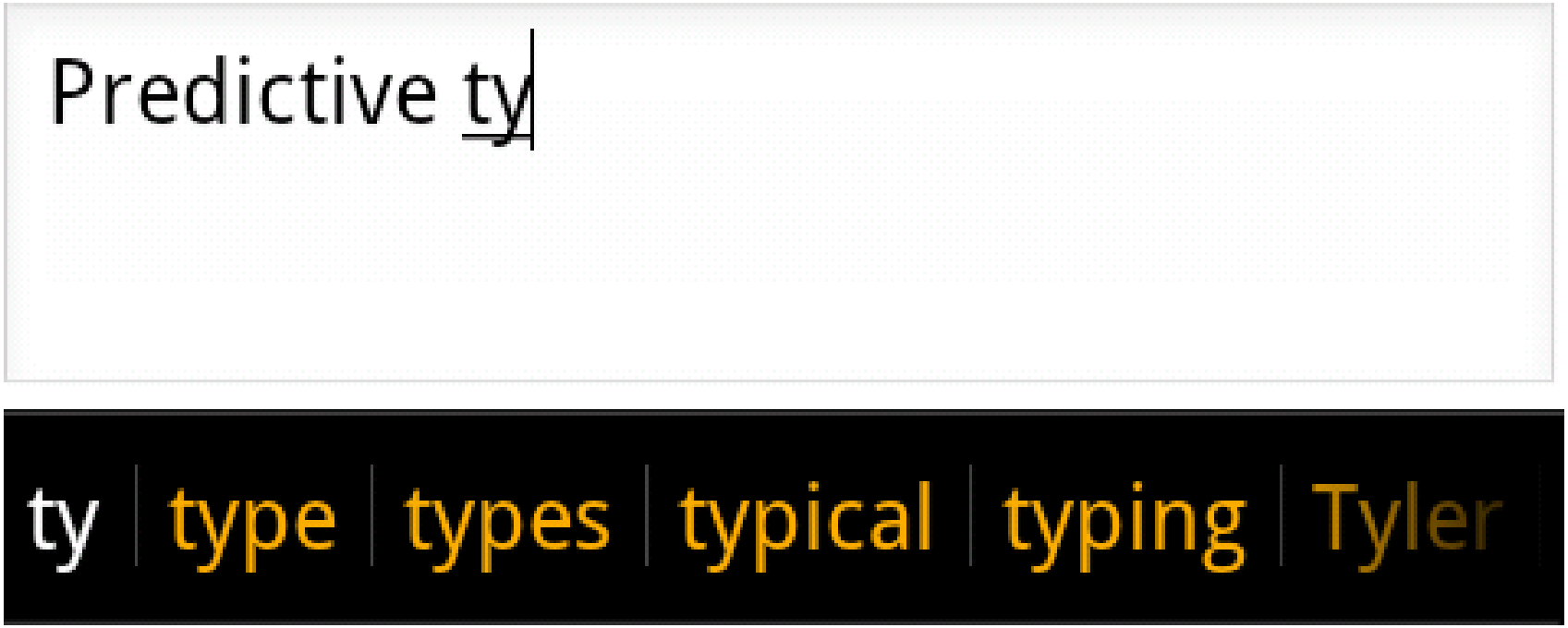
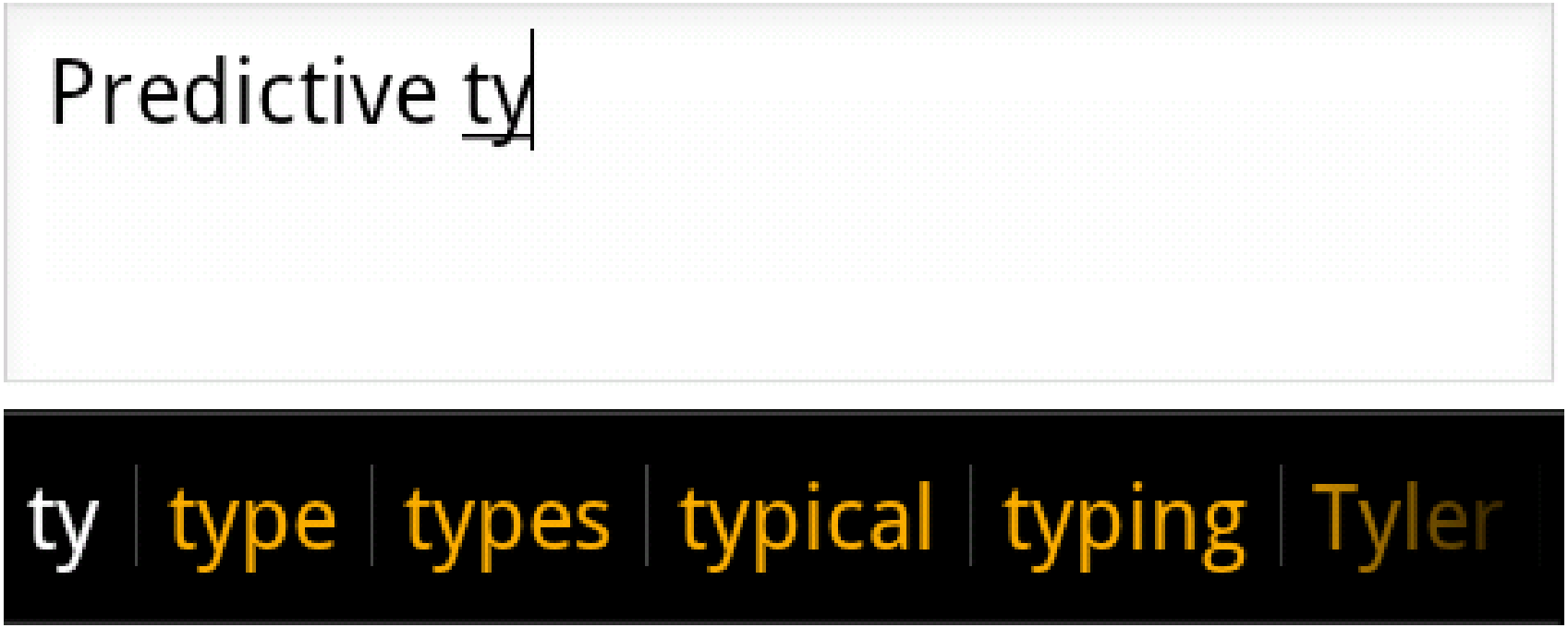
Language Modeling

Kenneth Heafield

Bloomberg


Predictive ty
ty
type
types
typical
typing
Tyler



$$p(\text{type} \mid \text{Predictive}) > p(\text{Tyler} \mid \text{Predictive})$$

Win or luse, it was a great game.

Win or lose, it were a great game.

Win or loose, it was a great game.

$$p(\text{lose} \mid \text{Win or}) \gg p(\text{loose} \mid \text{Win or})$$

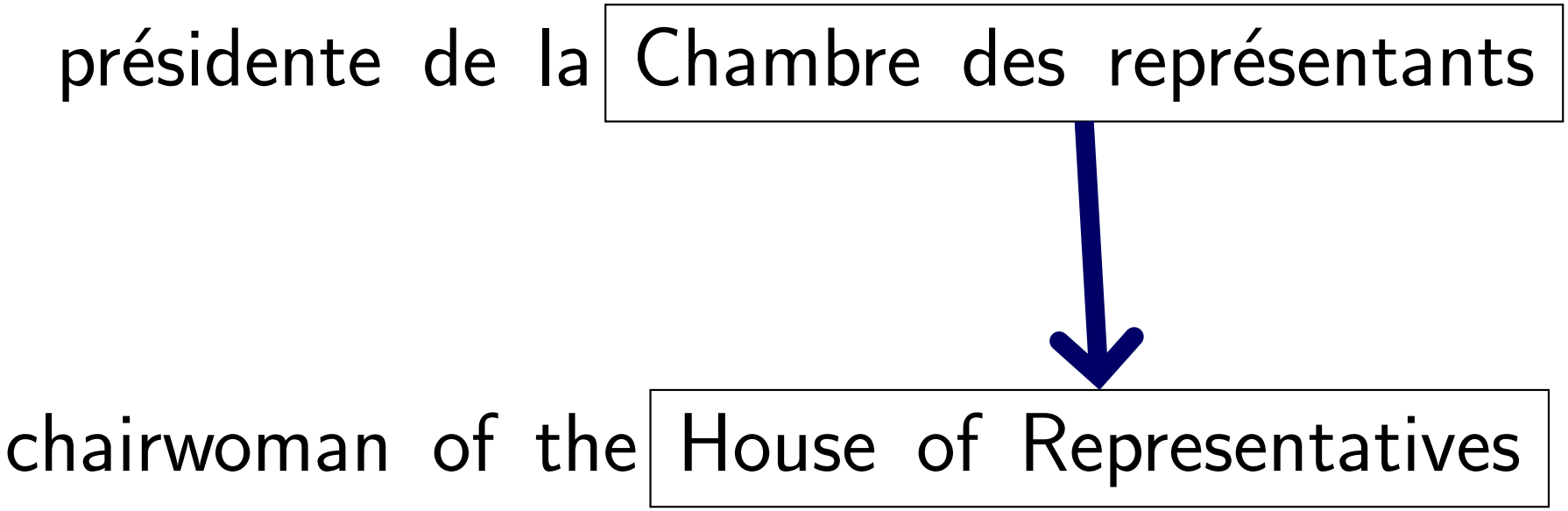
[Church et al, 2007]

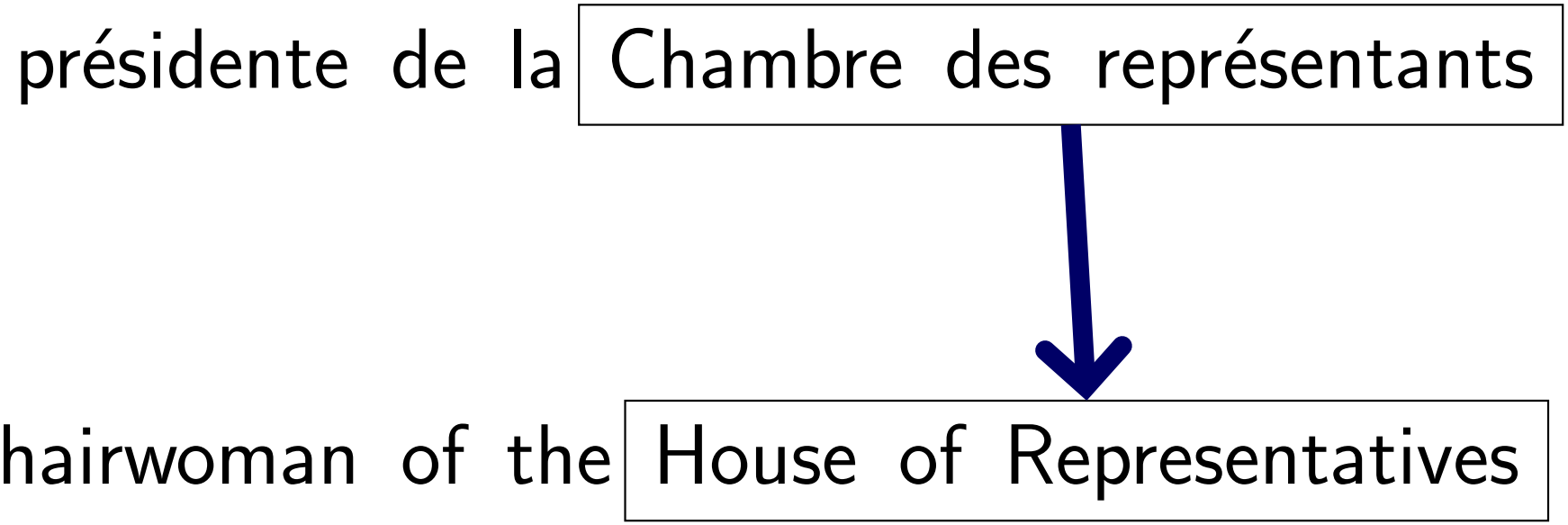
Chambre

Bedroom

présidente de la Chambre des représentants

chairwoman of the Bedroom of Representatives

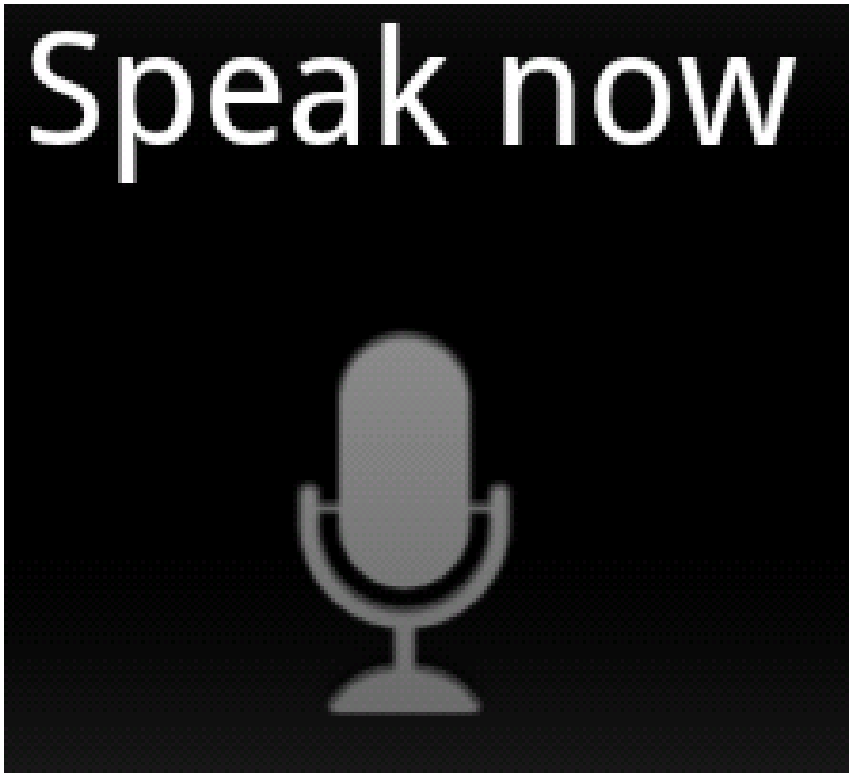

présidente de la
Chambre des représentants
chairwoman of the
House of Representatives



p(chairwoman of the House of Representatives)

$>$

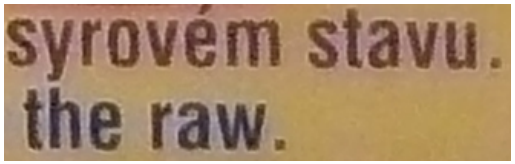
p(chairwoman of the Bedroom of Representatives)



$$p(\text{Another one bites the dust.}) > p(\text{Another one rides the bus.})$$

Prediction

Spelling



Translation

Speech

Essential Component: **Language Model**

$p(\text{in the raw}) = ?$

Language model: fluency of output

✗ Number of phrase pairs used to translate

✗ IBM model 1

✓ Length

✓ Ratio of letter “z” to letter “e”

Language model: fluency of output

✗ Number of phrase pairs used to translate

✗ IBM model 1

✓ Length

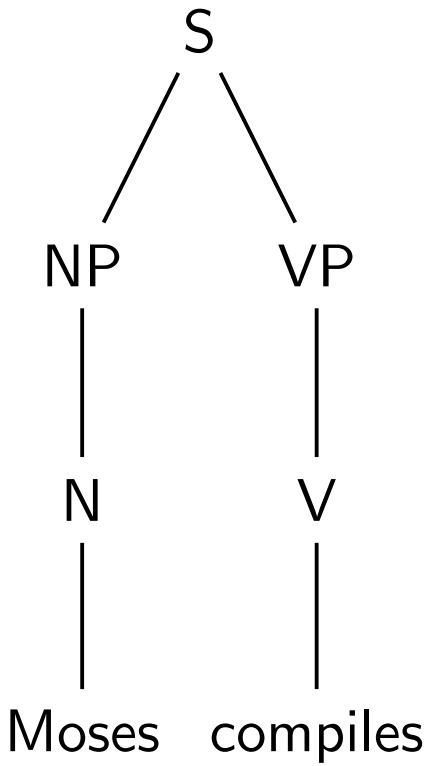
✓ Ratio of letter “z” to letter “e”

✓ Parsing

✓ **Sequence Models**

Parsing

$$p(\text{Moses compiles}) =$$



$$p(\text{S} \rightarrow \text{NP VP})$$

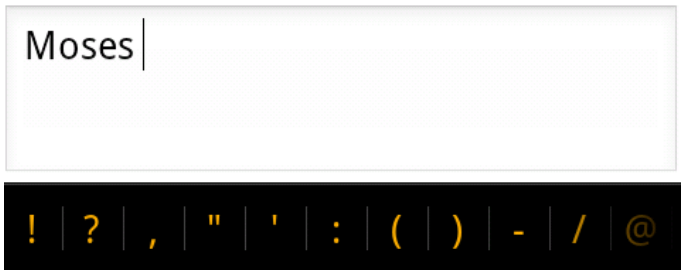
$$\cdot p(\text{NP} \rightarrow \text{N}) p(\text{VP} \rightarrow \text{V})$$

$$\cdot p(\text{N} \rightarrow \text{Moses}) p(\text{V} \rightarrow \text{compiles})$$

Sequence Models

Chain Rule

$$p(\text{Moses compiles}) = p(\text{Moses})\,p(\text{compiles} \mid \text{Moses})$$



Sequence Model

$$
\begin{aligned}
&\log p(\text{iran} \mid \text{<s>}) \\
&\log p(\text{is} \mid \text{<s> iran}) \\
&\log p(\text{one} \mid \text{<s> iran is}) \\
&\log p(\text{of} \mid \text{<s> iran is one}) \\
&\log p(\text{the} \mid \text{<s> iran is one of}) \\
&\log p(\text{few} \mid \text{<s> iran is one of the}) \\
&\log p(\text{countries} \mid \text{<s> iran is one of the few}) \\
&\log p(\text{.} \mid \text{<s> iran is one of the few countries}) \\
+\ &\log p(\text{</s>} \mid \text{<s> iran is one of the few countries .}) \\
\hline
=\ &\log p(\text{<s> iran is one of the few countries . </s>})
\end{aligned}
$$

Sequence Model

$$
\begin{array}{rlll}
 & \log p(\text{iran} & \mid \text{<s>} &) \\
 & \log p(\text{is} & \mid \text{<s> iran} &) \\
 & \log p(\text{one} & \mid \text{<s> iran is} &) \\
 & \log p(\text{of} & \mid \text{<s> iran is one} &) \\
 & \log p(\text{the} & \mid \text{<s> iran is one of} &) \\
 & \log p(\text{few} & \mid \text{<s> iran is one of the} &) \\
 & \log p(\text{countries} & \mid \text{<s> iran is one of the few} &) \\
 & \log p(\text{.} & \mid \text{<s> iran is one of the few countries} &) \\
+ & \log p(\text{</s>} & \mid \text{<s> iran is one of the few countries .} &) \\
\hline
= & \log p(\text{<s> iran is one of the few countries . </s>} & &)
\end{array}
$$

Explicit begin and end of sentence.

Sequence Model

$$
\begin{array}{lll}
 & \log p(\text{iran} & | \ \text{<s>} \ ) = \\
 & \log p(\text{is} & | \ \text{<s> iran} \ ) = \\
 & \log p(\text{one} & | \ \text{<s> iran is} \ ) = \\
 & \log p(\text{of} & | \ \text{<s> iran is one} \ ) = \\
 & \log p(\text{the} & | \ \text{<s> iran is one of} \ ) = \\
 & \log p(\text{few} & | \ \text{<s> iran is one of the} \ ) = \\
 & \log p(\text{countries} & | \ \text{<s> iran is one of the few} \ ) = \\
 & \log p(\text{.} & | \ \text{<s> iran is one of the few countries} \ ) = \\
+ & \log p(\text{</s>} & | \ \text{<s> iran is one of the few countries .}) = \\
\hline
= & \log p(\text{<s> iran is one of the few countries . </s>} &) =
\end{array}
$$

Where do these probabilities come from?

Probabilities from Text

Model

$p(\text{raw} \mid \text{in the})$

Estimating from Text

help in the search for an answer .
Copper burned in the raw wood .
put forward in the paper
Highs in the 50s to lower 60s .
⋮

$\Longrightarrow$

$$p(\text{raw} \mid \text{in the}) \approx \frac{1}{4}$$

Estimating from Text

help in the search for an answer .
Copper burned in the raw wood .
put forward in the paper
Highs in the 50s to lower 60s .
⋮

$\Longrightarrow$

$$p(\text{raw} \mid \text{in the}) \approx \frac{1}{4}$$
$$p(\text{Ugrasena} \mid \text{in the}) \approx 0$$

Estimating from Text

help in the search for an answer .
Copper burned in the raw wood .
put forward in the paper
Highs in the 50s to lower 60s .
⋮

$$\Longrightarrow \quad p(\text{raw} \mid \text{in the}) \approx \frac{1}{6}$$
$$p(\text{Ugrasena} \mid \text{in the}) \approx \frac{1}{1000}$$

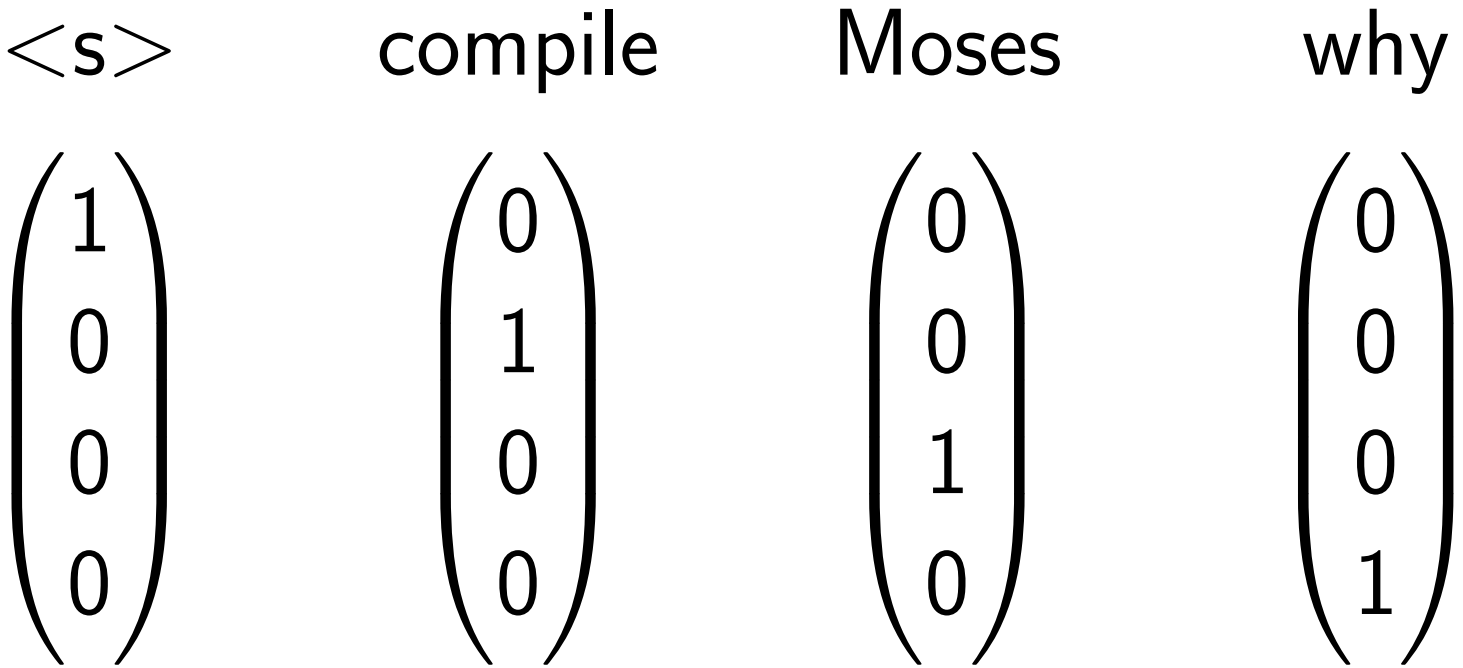
Smoothing

"in the Ugrasena" was not seen, but could happen.

1. **Neural Networks:**: classifier predicts next word
2. **Backoff**: maybe "the Ugrasena" was seen?

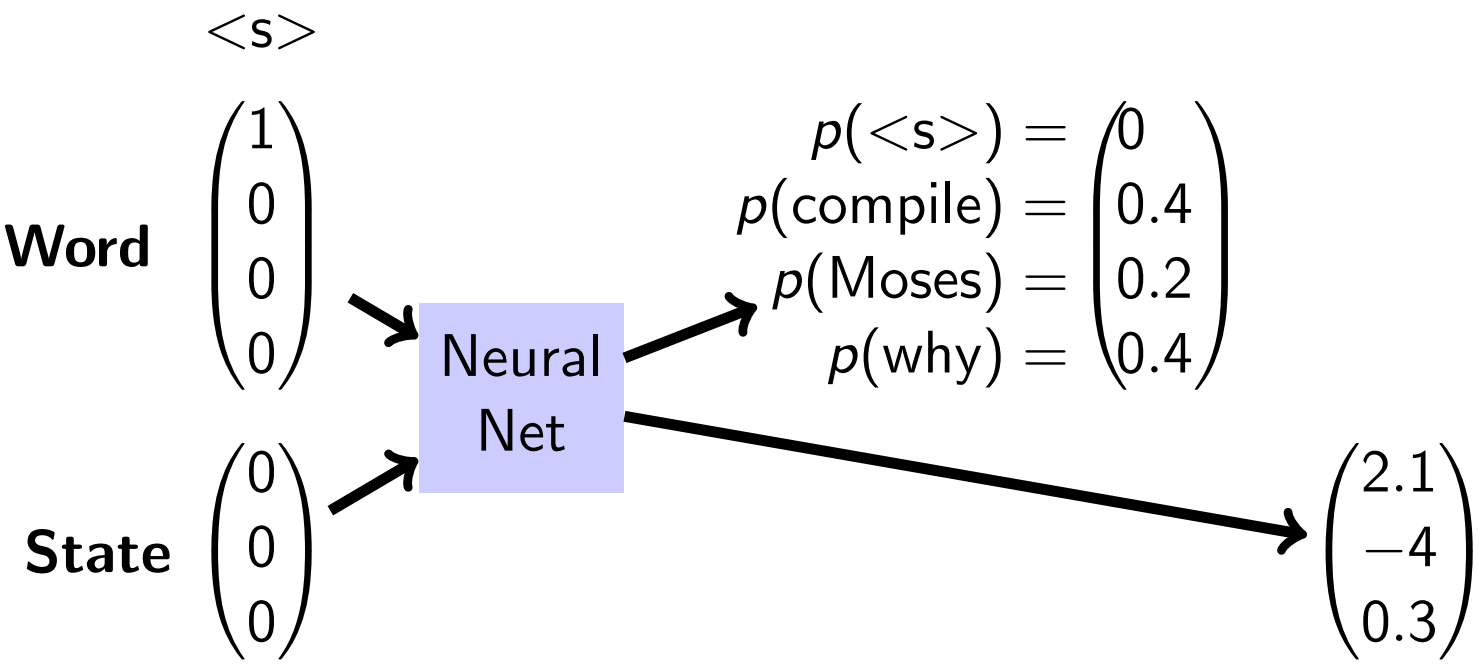
Language Modeling

Turning Words into Vectors

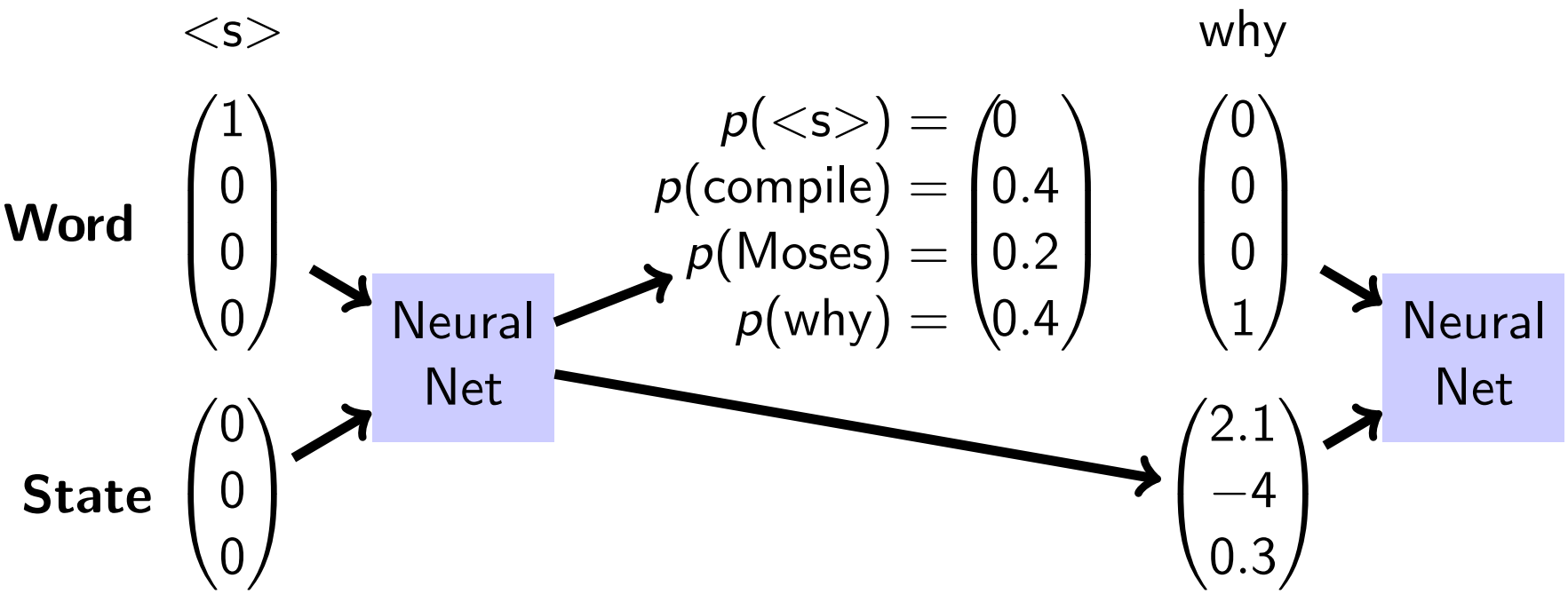


Assign each word a unique row.

Recurrent Neural Network



Recurrent Neural Network



Recurrent Neural Network Properties

Treat language modeling as a classification problem:
Predict the next word.

State uses the *entire* context back to the beginning:
Not forgetful like backoff.

Turning Words into Vectors

<s>	compile	Moses	why
$\begin{pmatrix} -3 \\ 1.5 \\ 6.2 \end{pmatrix}$	$\begin{pmatrix} 2.2 \\ 7.5 \\ -.8 \end{pmatrix}$	$\begin{pmatrix} -.1 \\ 0.8 \\ 9.1 \end{pmatrix}$	$\begin{pmatrix} 1.1 \\ 7.0 \\ -.2 \end{pmatrix}$

Vectors from a recurrent neural network.

Neural N-gram Models

$$p(\text{compile} \mid \text{Vector(why)}, \text{Vector(Moses)})$$

Vectors for context words

$\rightarrow$ neural network classifier

$\rightarrow$ probability distribution over words

Language Modeling

Backoff Smoothing

"in the Ugrasena" was not seen $\rightarrow$ try "the Ugrasena"

$$p(\text{Ugrasena} \mid \text{in the}) \approx p(\text{Ugrasena} \mid \text{the})$$

Backoff Smoothing

"in the Ugrasena" was not seen $\rightarrow$ try "the Ugrasena"

$$p(\text{Ugrasena} \mid \text{in the}) \approx p(\text{Ugrasena} \mid \text{the})$$

"the Ugrasena" was not seen $\rightarrow$ try "Ugrasena"

$$p(\text{Ugrasena} \mid \text{the}) \approx p(\text{Ugrasena})$$

Backoff Smoothing

"in the Ugrasena" was not seen $\rightarrow$ try "the Ugrasena"

$$p(\text{Ugrasena} \mid \text{in the}) = p(\text{Ugrasena} \mid \text{the})b(\text{in the})$$

"the Ugrasena" was not seen $\rightarrow$ try "Ugrasena"

$$p(\text{Ugrasena} \mid \text{the}) = p(\text{Ugrasena})b(\text{the})$$

Backoff b is a penalty for not matching context.

Example Language Model

Unigrams

Words	log p	log b
<s>	$-\infty$	-2.0
iran	-4.1	-0.8
is	-2.5	-1.4
one	-3.3	-0.9
of	-2.5	-1.1

Bigrams

Words	log p	log b
<s> iran	-3.3	-1.2
iran is	-1.7	-0.4
is one	-2.0	-0.9
one of	-1.4	-0.6

Trigrams

Words	log p
<s> iran is	-1.1
iran is one	-2.0
is one of	-0.3

Example Language Model

Unigrams

Words	log p	log b
<s>	$-\infty$	-2.0
iran	-4.1	-0.8
is	-2.5	-1.4
one	-3.3	-0.9
of	-2.5	-1.1

Bigrams

Words	log p	log b
<s> iran	-3.3	-1.2
iran is	-1.7	-0.4
is one	-2.0	-0.9
one of	-1.4	-0.6

Trigrams

Words	log p
<s> iran is	-1.1
iran is one	-2.0
is one of	-0.3

Query

$$\log p(\text{is} \mid \text{<s> iran}) = -1.1$$

Example Language Model

Unigrams

Words	log p	log b
<s>	$-\infty$	-2.0
iran	-4.1	-0.8
is	-2.5	-1.4
one	-3.3	-0.9
of	-2.5	-1.1

Bigrams

Words	log p	log b
<s> iran	-3.3	-1.2
iran is	-1.7	-0.4
is one	-2.0	-0.9
one of	-1.4	-0.6

Trigrams

Words	log p
<s> iran is	-1.1
iran is one	-2.0
is one of	-0.3

$$\text{Query}: \; p(\text{of} \mid \text{iran is})$$

$$\begin{aligned} \log p(\text{of}) & & -2.5 \\ \log b(\text{is}) & & -1.4 \\ \log b(\text{iran is}) & + & -0.4 \\ \hline \log p(\text{of} \mid \text{iran is}) & = & -4.3 \end{aligned}$$

Close words matter more.

Doubts

- Grammatical structure
- Topical coherence
- Words tend to repeat
- Tomorrow: Bonnie Webber on discourse

Alternative: skip over words in the context
[Pickhardt et al, ACL 2014]

Language Modeling

1 Smoothing
   Neural Networks
   Backoff
2 **Implementation Issues**

Stupid Backoff

1. Count *n*-grams offline

2. Compute pseudo-probabilities at runtime

[Brants et al, 2007]

Stupid Backoff

1. Count n-grams offline

$$\text{count}(w_1^n)$$

2. Compute pseudo-probabilities at runtime

$$\text{stupid}(w_n \mid w_1^{n-1}) = \begin{cases} \dfrac{\text{count}(w_1^n)}{\text{count}(w_1^{n-1})} & \text{if count}(w_1^n) > 0 \\ 0.4\text{stupid}(w_n \mid w_2^{n-1}) & \text{if count}(w_1^n) = 0 \end{cases}$$

Note: stupid does not sum to 1.

[Brants et al, 2007]

Counting *n*-grams

<s> Australia is one of the few

5-**gram**	**Count**
<s> Australia is one of	1
Australia is one of the	1
is one of the few	1

Hash table from *n*-gram to count.

Query

$$\text{stupid}(w_n \mid w_1^{n-1}) = \begin{cases} \dfrac{\text{count}(w_1^n)}{\text{count}(w_1^{n-1})} & \text{if count}(w_1^n) > 0 \\ 0.4\text{stupid}(w_n \mid w_2^{n-1}) & \text{if count}(w_1^n) = 0 \end{cases}$$

stupid(few | is one of the)

$$\text{count(is one of the few)} = 5 \checkmark$$

$$\text{count(is one of the)} = 12$$

Query

$$\text{stupid}(w_n \mid w_1^{n-1}) = \begin{cases} \dfrac{\text{count}(w_1^n)}{\text{count}(w_1^{n-1})} & \text{if count}(w_1^n) > 0 \\ 0.4\text{stupid}(w_n \mid w_2^{n-1}) & \text{if count}(w_1^n) = 0 \end{cases}$$

stupid(periwinkle | is one of the)

count(is one of the periwinkle) = 0 ✗
count(one of the periwinkle) = 0 ✗
count(of the periwinkle) = 0 ✗
count(the periwinkle) = 3 ✓

count(the) = 1000

What's Left?

- Hash table uses too much RAM
- Non-"stupid" smoothing methods (e.g. Kneser-Ney)

Save Memory: Forget Keys

Giant hash table with n-grams as keys and counts as values.

Replace the n-grams with 64-bit hashes:
Store hash(is one of) instead of "is one of".
Ignore collisions.

Save Memory: Forget Keys

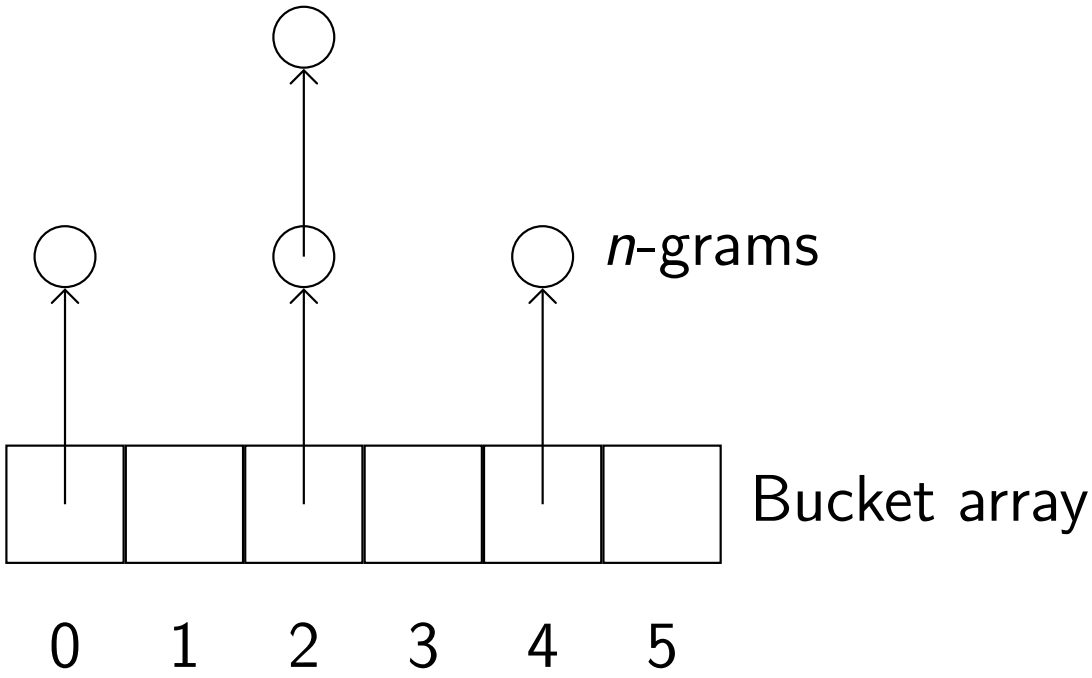
Giant hash table with n-grams as keys and counts as values.

Replace the n-grams with 64-bit hashes:
Store hash(is one of) instead of "is one of".
Ignore collisions.

Birthday attack: $\sqrt{2^{64}} = 2^{32}$.
$\Longrightarrow$ Low chance of collision until $\approx$ 4 billion entries.

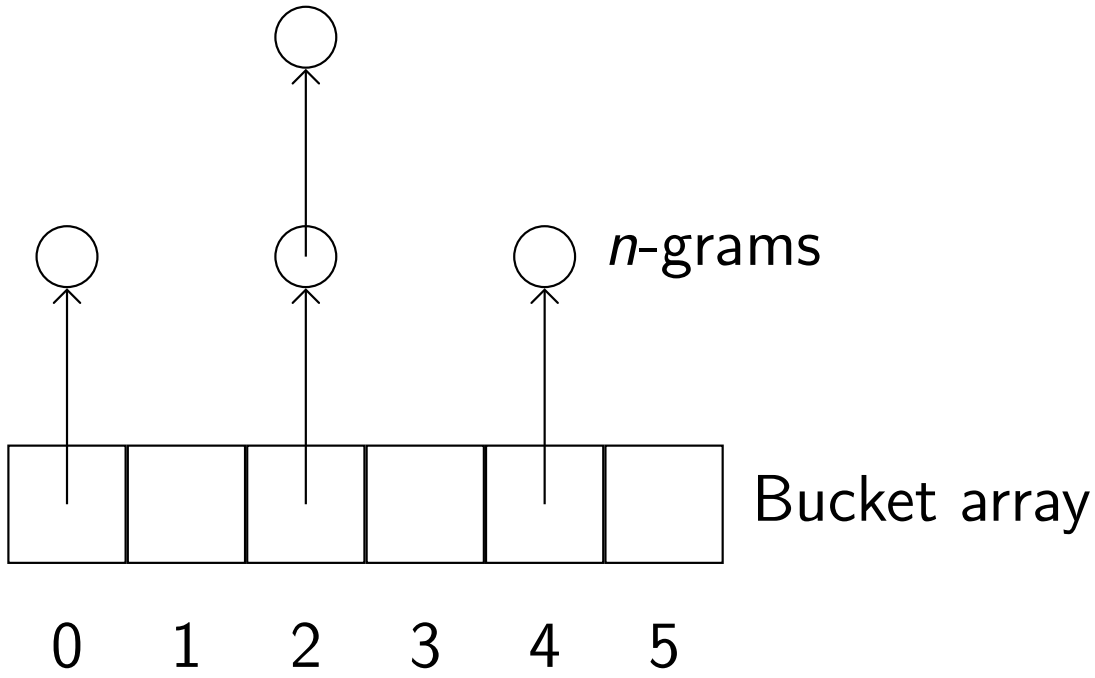
Default Hash Table

`boost::unordered_map` and `__gnu_cxx::hash_map`



Default Hash Table

`boost::unordered_map` and `__gnu_cxx::hash_map`



Lookup requires two random memory accesses.

Linear Probing Hash Table

- 1.5 buckets/entry (so buckets = 6).
- Ideal bucket = hash mod buckets.
- Resolve *bucket* collisions using the next free bucket.

Bigrams

Words	Ideal	Hash	Count
iran is	0	0x959e48455f4a2e90	3
		0x0	0
is one	2	0x186a7caef34acf16	5
one of	2	0xac66610314db8dac	2
<s> iran	4	0xf0ae9c2442c6920e	1
		0x0	0

Minimal Perfect Hash Table

Maps every n-gram to a unique integer $[0, |n - grams|)$
$\rightarrow$ Use these as array offsets.

Minimal Perfect Hash Table

Maps every n-gram to a unique integer $[0, |n - grams|)$
$\rightarrow$ Use these as array offsets.

Entries not in the model get assigned offsets
$\rightarrow$ Store a fingerprint of each n-gram

Minimal Perfect Hash Table

Maps every n-gram to a unique integer $[0, |n - grams|)$
$\rightarrow$ Use these as array offsets.

Low memory, but potential for false positives

Sorted Array

Sort n-grams, perform binary search.

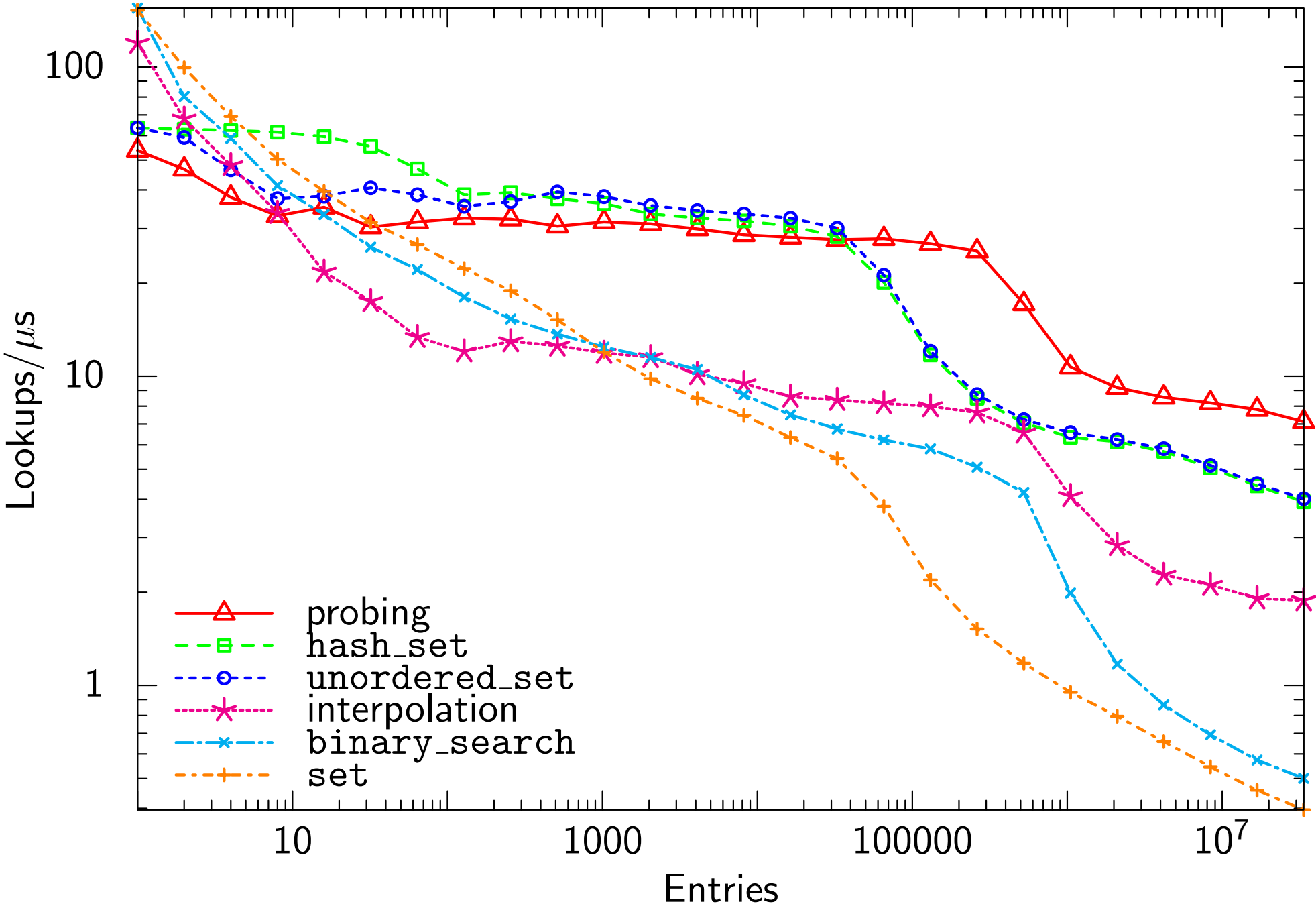
Binary search is $O(|n\text{-grams}| \log |n\text{-grams}|)$.

Sorted Array

Sort n-grams, perform binary search.

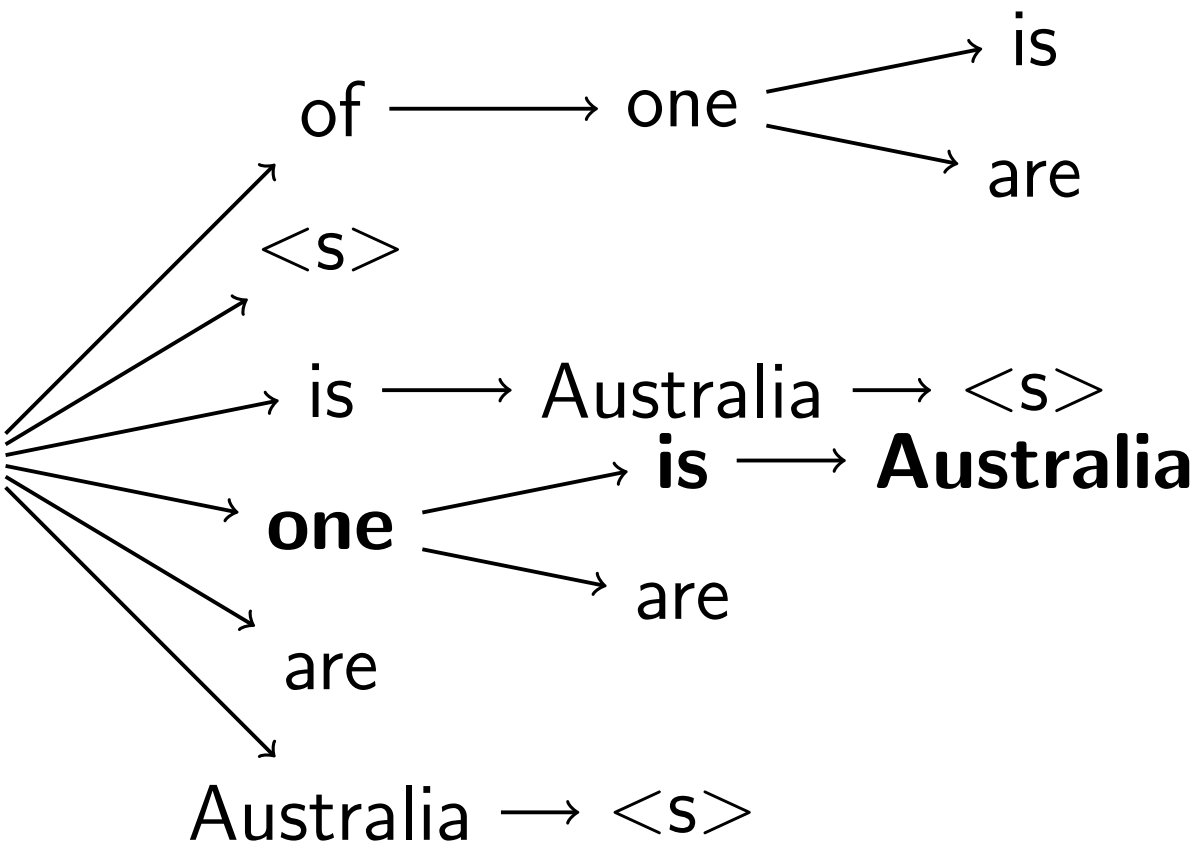
Binary search is $O(|n\text{-grams}| \log |n\text{-grams}|)$.

Interpolation search is $O(|n\text{-grams}| \log \log |n\text{-grams}|)$


100
10
1
Lookups/μs
10
1000
100000
10^7
Entries
probing
hash_set
unordered_set
interpolation
binary_search
set

Trie

Reverse *n*-grams, arrange in a trie.



Saving More RAM

- Quantization: store approximate values
- Collapse probability and backoff

Implementation Summary

Implementation involves sparse mapping

- Hash table
- Probing hash table
- Minimal perfect hash table
- Sorted array with binary or interpolation search

Conclusion

Language models measure fluency.

Neural networks and backoff are the dominant formalisms.

Efficient implementation needs good data structures.