Language Modeling

Kenneth Heafield

University of Edinburgh
Predictive typing
\[ p(\text{type} \mid \text{Predictive}) > p(\text{Tyler} \mid \text{Predictive}) \]
Win or lose, it was a great game.
Win or loose, it were a great game.
Win or lose, it was a great game.

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

[Church et al, 2007]
Heated indoor swimming pool
Introduction
Smoothing Kneser-Ney
Implementation
Conclusion

Chairwoman of the House of Representatives
Chairwoman of the Bedroom of Representatives

Bedroom
Chambre
présidente de la Chambre des représentants

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

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

chairwoman of the House of Representatives

\[ p(\text{chairwoman of the House of Representatives}) > p(\text{chairwoman of the Bedroom of Representatives}) \]
Another one bites the dust.

Another one rides the bus.

\[ p(\text{Another one bites the dust.}) > p(\text{Another one rides the bus.}) \]
Essential Component: Language Model

\[ p(\text{in the raw}) = ? \]
Language model: fluency of output

- How well it translates the source
- Ratio to source sentence

- Length
- Ratio of letter “z” to letter “e”
Language model: fluency of output

✗ How well it translates the source
✗ Ratio to source sentence

✓ Length
✓ Ratio of letter “z” to letter “e”
✓ Parsing
✓ Sequence Models
\[ p(\text{Moses compiles}) = \]

\[ p(\text{S} \rightarrow \text{NP VP}) \]

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

\[ \cdot p(\text{N} \rightarrow \text{Moses}) p(\text{V} \rightarrow \text{compiles}) \]
Sequence Models

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

Chain Rule

Moses

! ? , " ' : ( ) - / @
Sequence Model

\[
\begin{align*}
\log p(\text{iran} & | <s>) = -3.33437 \\
\log p(\text{is} & | <s> \text{ iran}) = -1.05931 \\
\log p(\text{one} & | <s> \text{ iran is}) = -1.80743 \\
\log p(\text{of} & | <s> \text{ iran is one}) = -0.03705 \\
\log p(\text{the} & | <s> \text{ iran is one of}) = -0.08317 \\
\log p(\text{few} & | <s> \text{ iran is one of the}) = -1.20788 \\
\log p(\text{countries} & | <s> \text{ iran is one of the few}) = -1.62030 \\
\log p(\text{.} & | <s> \text{ iran is one of the few countries}) = -2.60261 \\
+ \log p(</s>) & | <s> \text{ iran is one of the few countries .}) = -0.04688 \\
\end{align*}
\]

= \log p(<s> \text{ iran is one of the few countries .} </s>)

Where do these probabilities come from?
\[
\begin{align*}
\log p(\text{iran} | <s>) &= -3.33437 \\
\log p(\text{is} | <s> \text{iran}) &= -1.05931 \\
\log p(\text{one} | <s> \text{iran is}) &= -1.80743 \\
\log p(\text{of} | <s> \text{iran is one}) &= -0.03705 \\
\log p(\text{the} | <s> \text{iran is one of}) &= -0.08317 \\
\log p(\text{few} | <s> \text{iran is one of the}) &= -1.20788 \\
\log p(\text{countries} | <s> \text{iran is one of the few}) &= -1.62030 \\
\log p(\text{.} | <s> \text{iran is one of the few countries}) &= -2.60261 \\
+ \log p(</s>) &= -11.79900 \\
= \log p(<s> \text{iran is one of the few countries} . </s>) &= -0.04688
\end{align*}
\]

Explicit begin and end of sentence.
Sequence Model

\[
\begin{align*}
\log p(\text{iran} \mid <s>) &= -3.33437 \\
\log p(\text{is} \mid <s> \text{ iran}) &= -1.05931 \\
\log p(\text{one} \mid <s> \text{ iran is}) &= -1.80743 \\
\log p(\text{of} \mid <s> \text{ iran is one}) &= -0.03705 \\
\log p(\text{the} \mid <s> \text{ iran is one of}) &= -0.08317 \\
\log p(\text{few} \mid <s> \text{ iran is one of the}) &= -1.20788 \\
\log p(\text{countries} \mid <s> \text{ iran is one of the few}) &= -1.62030 \\
\log p(\text{.} \mid <s> \text{ iran is one of the few countries}) &= -2.60261 \\
+ \log p(<\!/s>) \mid <s> \text{ iran is one of the few countries .} &= -0.04688 \\
\end{align*}
\]

\[
= \log p(<s> \text{ iran is one of the few countries .} <\!/s>) = -11.79900
\]

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.

\[
p(\text{raw} \mid \text{in the}) \approx \frac{1}{4}
\]
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.

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

\[ p(\text{raw} \mid \text{in the}) \approx \frac{1}{6} \]
\[ p(\text{Ugrasena} \mid \text{in the}) \approx \frac{1}{1000} \]
Problem

“in the Ugrasena” was not seen, but could happen.

\[ p(\text{Ugrasena} \mid \text{in the}) = \frac{\text{count(\text{in the Ugrasena})}}{\text{count(\text{in the})}} = 0? \]
Problem

“in the Ugrasena” was not seen, but could happen.

\[ p(Ugrasena \mid \text{in the}) = \frac{\text{count}(\text{in the Ugrasena})}{\text{count}(\text{in the})} = 0? \]

\[ = \frac{\text{count}(\text{the Ugrasena})}{\text{count}(\text{the})} = 2.07 \cdot 10^{-9} \]
Problem

“in the Ugrasena” was not seen, but could happen.

\[ p(\text{Ugrasena} \mid \text{in the}) = \frac{\text{count(\text{in the Ugrasena})}}{\text{count(\text{in the})}} = 0? \]
\[ = \frac{\text{count(\text{the Ugrasena})}}{\text{count(\text{the})}} = 2.07 \cdot 10^{-9} \]

Stupid Backoff: Drop context until count is non-zero
[Brants et al, 2007]

Can we be less stupid?
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

1 Smoothing
   Neural Networks
   Backoff
2 Kneser-Ney Smoothing
3 Implementation
Assign each word a unique row.
Recurrent Neural Network

Word

State

Neural Net

\[
p(<s>) = \begin{pmatrix} 0 \\ 0.4 \\ 0.2 \\ 0.4 \end{pmatrix}
\]

\[
p(\text{in}) = \begin{pmatrix} 0 \\ 2.1 \\ -4 \end{pmatrix}
\]

\[
p(\text{the}) = \begin{pmatrix} 0 \end{pmatrix}
\]

\[
p(\text{raw}) = \begin{pmatrix} 0.3 \end{pmatrix}
\]
Recurrent Neural Network

\[
p(<s>) = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}
\]
\[
p(in) = \begin{pmatrix} 0.4 \\ 0 \\ 0 \end{pmatrix}
\]
\[
p(the) = \begin{pmatrix} 0.2 \\ 0 \\ 0 \end{pmatrix}
\]
\[
p(raw) = \begin{pmatrix} 0.4 \\ 0 \\ 1 \end{pmatrix}
\]

\[
praw = \begin{pmatrix} 2.1 \\ -4 \\ 0.3 \end{pmatrix}
\]
Treat language modeling as a classification problem: Predict the next word.

**State** uses the *entire* context back to the beginning.
Turning Words into Vectors

<table>
<thead>
<tr>
<th></th>
<th>in</th>
<th>the</th>
<th>raw</th>
</tr>
</thead>
<tbody>
<tr>
<td>⟨s⟩</td>
<td>(−3)</td>
<td>(2.2)</td>
<td>(−0.1)</td>
</tr>
<tr>
<td></td>
<td>(1.5)</td>
<td>(7.5)</td>
<td>(0.8)</td>
</tr>
<tr>
<td></td>
<td>(6.2)</td>
<td>(−0.8)</td>
<td>(9.1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(1.1)</td>
<td>(7.0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(−0.2)</td>
</tr>
</tbody>
</table>

Vectors from a recurrent neural network
... or your favorite ACL paper.
Neural N-gram Models

\[ p(\text{raw} \mid \text{Vector(in)}, \text{Vector(the)}) \]

Vectors for context words
\[ \rightarrow \] neural network classifier
\[ \rightarrow \] probability distribution over words
Language Modeling

1. Smoothing
   Neural Networks
   Backoff
2. Kneser-Ney Smoothing
3. Implementation
Backoff Smoothing

“in the Ugrasena” was not seen $\rightarrow$ try “the Ugrasena”

$p(Ugrasena \mid \text{in the}) \approx p(Ugrasena \mid \text{the})$
“in the Ugrasena” was not seen → try “the Ugrasena”
\[ p(\text{Ugrasena} \mid \text{in the}) \approx p(\text{Ugrasena} \mid \text{the}) \]

“the Ugrasena” was not seen → try “Ugrasena”
\[ p(\text{Ugrasena} \mid \text{the}) \approx p(\text{Ugrasena}) \]
Backoff Smoothing

“in the Ugrasena” was not seen → try “the Ugrasena”

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

“the Ugrasena” was not seen → try “Ugrasena”

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

Backoff \( b \) is a penalty for not matching context.
# Example Language Model

<table>
<thead>
<tr>
<th>Unigrams</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Words</td>
<td>log $p$</td>
<td>log $b$</td>
<td></td>
</tr>
<tr>
<td>$&lt;s&gt;$</td>
<td>$-\infty$</td>
<td>$-2.0$</td>
<td></td>
</tr>
<tr>
<td>iran</td>
<td>$-4.1$</td>
<td>$-0.8$</td>
<td></td>
</tr>
<tr>
<td>is</td>
<td>$-2.5$</td>
<td>$-1.4$</td>
<td></td>
</tr>
<tr>
<td>one</td>
<td>$-3.3$</td>
<td>$-0.9$</td>
<td></td>
</tr>
<tr>
<td>of</td>
<td>$-2.5$</td>
<td>$-1.1$</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bigrams</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Words</td>
<td>log $p$</td>
<td>log $b$</td>
<td></td>
</tr>
<tr>
<td>$&lt;s&gt;$ iran</td>
<td>$-3.3$</td>
<td>$-1.2$</td>
<td></td>
</tr>
<tr>
<td>iran is</td>
<td>$-1.7$</td>
<td>$-0.4$</td>
<td></td>
</tr>
<tr>
<td>is one</td>
<td>$-2.0$</td>
<td>$-0.9$</td>
<td></td>
</tr>
<tr>
<td>one of</td>
<td>$-1.4$</td>
<td>$-0.6$</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Trigrams</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Words</td>
<td>log $p$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$&lt;s&gt;$ iran is</td>
<td>$-1.1$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>iran is one</td>
<td>$-2.0$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>is one of</td>
<td>$-0.3$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Example Language Model

### Unigrams
<table>
<thead>
<tr>
<th>Words</th>
<th>log $p$</th>
<th>log $b$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$&lt;$s$&gt;$</td>
<td>$-\infty$</td>
<td>-2.0</td>
</tr>
<tr>
<td>iran</td>
<td>-4.1</td>
<td>-0.8</td>
</tr>
<tr>
<td>is</td>
<td>-2.5</td>
<td>-1.4</td>
</tr>
<tr>
<td>one</td>
<td>-3.3</td>
<td>-0.9</td>
</tr>
<tr>
<td>of</td>
<td>-2.5</td>
<td>-1.1</td>
</tr>
</tbody>
</table>

### Bigrams
<table>
<thead>
<tr>
<th>Words</th>
<th>log $p$</th>
<th>log $b$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$&lt;$s$&gt;$ iran</td>
<td>-3.3</td>
<td>-1.2</td>
</tr>
<tr>
<td>iran is</td>
<td>-1.7</td>
<td>-0.4</td>
</tr>
<tr>
<td>is one</td>
<td>-2.0</td>
<td>-0.9</td>
</tr>
<tr>
<td>one of</td>
<td>-1.4</td>
<td>-0.6</td>
</tr>
</tbody>
</table>

### Trigrams
<table>
<thead>
<tr>
<th>Words</th>
<th>log $p$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$&lt;$s$&gt;$ iran is</td>
<td>-1.1</td>
</tr>
<tr>
<td>iran is one</td>
<td>-2.0</td>
</tr>
<tr>
<td>is one of</td>
<td>-0.3</td>
</tr>
</tbody>
</table>

### Query

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

<table>
<thead>
<tr>
<th></th>
<th>Unigrams</th>
<th></th>
<th>Bigrams</th>
<th></th>
<th>Trigrams</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Words</td>
<td>log ( p )</td>
<td>log ( b )</td>
<td>Words</td>
<td>log ( p )</td>
<td>log ( b )</td>
<td>Words</td>
</tr>
<tr>
<td>&lt;s&gt;</td>
<td>(-\infty)</td>
<td>(-2.0)</td>
<td>&lt;s&gt;</td>
<td>(-3.3)</td>
<td>(-1.2)</td>
<td>&lt;s&gt;</td>
</tr>
<tr>
<td>iran</td>
<td>(-4.1)</td>
<td>(-0.8)</td>
<td>iran is</td>
<td>(-1.7)</td>
<td>(-0.4)</td>
<td>iran is one</td>
</tr>
<tr>
<td>is</td>
<td>(-2.5)</td>
<td>(-1.4)</td>
<td>is one</td>
<td>(-2.0)</td>
<td>(-0.9)</td>
<td>is one of</td>
</tr>
<tr>
<td>one</td>
<td>(-3.3)</td>
<td>(-0.9)</td>
<td>one of</td>
<td>(-1.4)</td>
<td>(-0.6)</td>
<td></td>
</tr>
<tr>
<td>of</td>
<td>(-2.5)</td>
<td>(-1.1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

\[
\log p(\text{of}) = -2.5 \\
\log b(\text{is}) = -1.4 \\
\log b(\text{iran is}) = -0.4 \\
\log p(\text{of} \mid \text{iran is}) = -4.3
\]
Close words matter more.

Though long-distance matters:
  Grammatical structure
  Topical coherence
  Words tend to repeat
  Cross-sentence dependencies

Alternative: skip over words in the context

[Pickhardt et al, ACL 2014]
Language Modeling

1 Smoothing
Neural Networks
Backoff

2 Kneser-Ney Smoothing

3 Implementation
Where do $p$ and $b$ come from?

Text!

Kneser-Ney
Witten-Bell
Good-Turing
Common high-quality smoothing

1. Adjust
2. Normalize
3. Interpolate
Adjusted counts are:

**Trigrams** Count in the text.

**Others** Number of unique words to the left.

Lower orders are used when a trigram did not match.
How freely does the text associate with new words?
Adjusted counts are:

**Trigrams** Count in the text.

**Others** Number of unique words to the left.

Lower orders are used when a trigram did not match.

How freely does the text associate with new words?

---

<table>
<thead>
<tr>
<th>Input Trigram</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>are one of</td>
<td>1</td>
</tr>
<tr>
<td>is one of</td>
<td>5</td>
</tr>
<tr>
<td>are two of</td>
<td>3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Output 1-gram Count</th>
<th>Adjusted</th>
</tr>
</thead>
<tbody>
<tr>
<td>one of</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Output 2-gram</th>
<th>Adjusted</th>
</tr>
</thead>
<tbody>
<tr>
<td>one of</td>
<td>2</td>
</tr>
<tr>
<td>two of</td>
<td>1</td>
</tr>
</tbody>
</table>
Discounting and Normalization

\[ \text{pseudo}(w_n | w_1^{n-1}) = \frac{\text{adjusted}(w_1^n) - \text{discount}_n(\text{adjusted}(w_1^n))}{\sum_x \text{adjusted}(w_1^{n-1}x)} \]

- Save mass for unseen events
- Normalize
Discounting and Normalization

\[ \text{pseudo}(w_n|w_1^{n-1}) = \frac{\text{adjusted}(w_1^n) - \text{discount}_n(\text{adjusted}(w_1^n))}{\sum_x \text{adjusted}(w_1^{n-1}x)} \]

Save mass for unseen events

Normalize

Input

<table>
<thead>
<tr>
<th>3-gram</th>
<th>Adjusted</th>
</tr>
</thead>
<tbody>
<tr>
<td>are one of</td>
<td>1</td>
</tr>
<tr>
<td>are one that</td>
<td>2</td>
</tr>
<tr>
<td>is one of</td>
<td>5</td>
</tr>
</tbody>
</table>

Output

<table>
<thead>
<tr>
<th>3-gram</th>
<th>Pseudo</th>
</tr>
</thead>
<tbody>
<tr>
<td>are one of</td>
<td>0.26</td>
</tr>
<tr>
<td>are one that</td>
<td>0.47</td>
</tr>
<tr>
<td>is one of</td>
<td>0.62</td>
</tr>
</tbody>
</table>
Interpolate: Sparsity vs. Specificity

Interpolate unigrams with the uniform distribution.

\[ p\text{(of)} = \text{pseudo(of)} + \text{backoff}(\epsilon) \frac{1}{|\text{vocabulary}|} \]
Interpolate: Sparsity vs. Specificity

Interpolate unigrams with the uniform distribution,

\[ p(\text{of}) = \text{pseudo}(\text{of}) + \text{backoff}(\epsilon) \frac{1}{|\text{vocabulary}|} \]

Interpolate bigrams with unigrams, etc.

\[ p(\text{of}|\text{one}) = \text{pseudo}(\text{of} | \text{one}) + \text{backoff}(\text{one}) p(\text{of}) \]
Interpolate: Sparsity vs. Specificity

Interpolate unigrams with the uniform distribution,
\[ p(\text{of}) = \text{pseudo(\text{of})} + \text{backoff(\(\epsilon\))} \frac{1}{|\text{vocabulary}|} \]

Interpolate bigrams with unigrams, etc.
\[ p(\text{of|one}) = \text{pseudo(\text{of | one})} + \text{backoff(\text{one})} p(\text{of}) \]

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>(n)-gram</td>
<td>pseudo</td>
</tr>
<tr>
<td>of</td>
<td>0.1</td>
</tr>
<tr>
<td>one of</td>
<td>0.2</td>
</tr>
<tr>
<td>are one of</td>
<td>0.4</td>
</tr>
</tbody>
</table>
Kneser-Ney Intuition

Adjust Measure association with new words.
Normalize Leave space for unseen events.
Interpolate Handle sparsity.

How do we implement it?
Language Modeling

1 Smoothing
   Neural Networks
   Backoff
2 Kneser-Ney Smoothing
3 Implementation
“LM queries often account for more than 50% of the CPU”
[Green et al, WMT 2014]

500 billion entries in my largest model

Need speed and memory efficiency
Counting $n$-grams

$<s>\text{ Australia is one of the few}$

<table>
<thead>
<tr>
<th>5-gram</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>$&lt;s&gt;\text{ Australia is one of}$</td>
<td>1</td>
</tr>
<tr>
<td>Australia is one of the</td>
<td>1</td>
</tr>
<tr>
<td>is one of the few</td>
<td>1</td>
</tr>
</tbody>
</table>

Hash table?
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?
Runs out of RAM.
Spill to Disk When RAM Runs Out

- Text
- Hash Table
- File
Split Data

Text

Hash Table

File

Hash Table

File

Introduction

Smoothing

Kneser-Ney

Implementation

Conclusion
Split and Merge

Text

Hash Table

Sort

File

Merge Sort
Training Problem:
Batch process large number of records.

Solution: Split/merge
Stupid backoff in one pass
Kneser-Ney in three passes
Training Problem:
Batch process large number of records.

Solution: Split/merge
- Stupid backoff in one pass
- Kneser-Ney in three passes

Training is designed for mutable batch access.
What about queries?
stupid($w_n \mid w_{n-1}^n$) = \begin{cases} \frac{\text{count}(w_1^n)}{\text{count}(w_{n-1}^{n-1})} & \text{if count}(w_1^n) > 0 \\ 0.4 \text{stupid}(w_n \mid w_{n-1}^2) & \text{if count}(w_1^n) = 0 \end{cases}

\text{stupid(few} \mid \text{is one of the})

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

\text{count(is one of the} = 12
stupid\((w_n \mid w_1^{n-1})\) = \begin{cases} \frac{\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}

\text{stupid}(\text{periwinkle} \mid \text{is one of the})

\text{count(is one of the periwinkle)} = 0 \times
\text{count(one of the periwinkle)} = 0 \times
\text{count(of the periwinkle)} = 0 \times
\text{count(the periwinkle)} = 3 \checkmark
\text{count(the)} = 1000
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.
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: $\sqrt{2^{64}} = 2^{32}$.

$\implies$ Low chance of collision until $\approx$ 4 billion entries.
Default Hash Table

boost::unordered_map and __gnu_cxx::hash_map

Bucket array

$n$-grams
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.

<table>
<thead>
<tr>
<th>Words</th>
<th>Ideal</th>
<th>Hash</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>iran is</td>
<td>0</td>
<td>0x959e48455f4a2e90</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x0</td>
<td>0</td>
</tr>
<tr>
<td>is one</td>
<td>2</td>
<td>0x186a7caef34acf16</td>
<td>5</td>
</tr>
<tr>
<td>one of</td>
<td>2</td>
<td>0xac66610314db8dac</td>
<td>2</td>
</tr>
<tr>
<td>&lt;s&gt; iran</td>
<td>4</td>
<td>0xf0ae9c2442c6920e</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x0</td>
<td>0</td>
</tr>
</tbody>
</table>
Minimal Perfect Hash Table

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

Entries not in the model get assigned offsets

Introduction
Smoothing
Kneser-Ney
Implementation
Conclusion
Minimal Perfect Hash Table

Maps every $n$-gram to a unique integer $[0, |n - \text{grams}|)$
   $\rightarrow$ 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
Less Memory: Sorted Array

Look up “zebra” in a dictionary.

Binary search
Open in the middle. $O(n \log n)$ time.

Interpolation search
Open near the end. $O(n \log \log n)$ time.
Reverse $n$-grams, arrange in a trie.

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
Language models measure fluency.

Neural networks and backoff are the dominant formalisms.

Efficient implementation needs good data structures.