Language Modeling

Kenneth Heafield

Bloomberg
Predictive typing
\[ p(\text{type} \mid \text{Predictive}) > p(\text{Tyler} \mid \text{Predictive}) \]
Win or lose, 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
présidente de la Chambre des représentants

chairwoman of the House of Representatives

\[ p(\text{chairwoman of the House of Representatives}) \]
\[ > \]
\[ p(\text{chairwoman of the Bedroom of Representatives}) \]
\[ p(\text{Another one bites the dust.}) > p(\text{Another one rides the bus.}) \]
Essential Component: **Language Model**

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

✗ 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

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

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

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

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

Chain Rule

\[ p(\text{Moses compiles}) = p(\text{Moses})p(\text{compiles | Moses}) \]
Sequence Model

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

\[
\begin{align*}
\log p(\text{iran}) & | <s> \\
\log p(\text{is}) & | <s> \text{ iran} \\
\log p(\text{one}) & | <s> \text{ iran is} \\
\log p(\text{of}) & | <s> \text{ iran is one} \\
\log p(\text{the}) & | <s> \text{ iran is one of} \\
\log p(\text{few}) & | <s> \text{ iran is one of the} \\
\log p(\text{countries}) & | <s> \text{ iran is one of the few} \\
\log p(\text{.}) & | <s> \text{ iran is one of the few countries} \\
+ \log p(<{/s}> ) & | <s> \text{ iran is one of the few countries .} \\
= \log p( <s> \text{ iran is one of the few countries .} <{/s}> ) 
\end{align*}
\]

Explicit begin and end of sentence.
Where do these probabilities come from?
Probabilities from Text

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

Model
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.

\[
p(\text{raw} \mid \text{in the}) \approx \frac{1}{6}
\]
\[
p(\text{Ugrasena} \mid \text{in the}) \approx \frac{1}{1000}
\]
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 Implementation Issues
### Turning Words into Vectors

Assign each word a unique row.

<table>
<thead>
<tr>
<th></th>
<th>compile</th>
<th>Moses</th>
<th>why</th>
</tr>
</thead>
<tbody>
<tr>
<td>$&lt;s&gt;$</td>
<td>$&lt;\text{compile}&gt;$</td>
<td>$&lt;\text{Moses}&gt;$</td>
<td>$&lt;\text{why}&gt;$</td>
</tr>
</tbody>
</table>
| \[
\begin{pmatrix}
1 \\
0 \\
0 \\
0
\end{pmatrix}
\] | \[
\begin{pmatrix}
0 \\
1 \\
0 \\
0
\end{pmatrix}
\] | \[
\begin{pmatrix}
0 \\
0 \\
1 \\
0
\end{pmatrix}
\] | \[
\begin{pmatrix}
0 \\
0 \\
0 \\
1
\end{pmatrix}
\] |
Recurrent Neural Network

\[
\begin{bmatrix}
1 \\
0 \\
0 \\
0 \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
0 \\
0.4 \\
0.2 \\
0.4 \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
2.1 \\
-4 \\
0.3 \\
\end{bmatrix}
\]
Recurrent Neural Network

\[
\begin{bmatrix}
1 \\
0 \\
0 \\
0
\end{bmatrix}
\begin{bmatrix}
< s > \\
\end{bmatrix}
\]

\[
p(<s>) = \begin{bmatrix} 0 \\ 0 \end{bmatrix}
\]

\[
p(\text{compile}) = \begin{bmatrix} 0.4 \\ 0 \end{bmatrix}
\]

\[
p(\text{Moses}) = \begin{bmatrix} 0.2 \\ 0 \end{bmatrix}
\]

\[
p(\text{why}) = \begin{bmatrix} 0.4 \\ 1 \end{bmatrix}
\]

\[
\begin{bmatrix}
0 \\
0 \\
2.1 \\
-4 \\
0.3
\end{bmatrix}
\]

\[
\begin{bmatrix}
\text{Neural Net} \\
\text{Neural Net}
\end{bmatrix}
\]
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

<table>
<thead>
<tr>
<th>$&lt;s&gt;$</th>
<th>compile</th>
<th>Moses</th>
<th>why</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(-3)$</td>
<td>$(2.2)$</td>
<td>$(-.1)$</td>
<td>$(1.1)$</td>
</tr>
<tr>
<td>$1.5$</td>
<td>$7.5$</td>
<td>$0.8$</td>
<td>$7.0$</td>
</tr>
<tr>
<td>$6.2$</td>
<td>$-.8$</td>
<td>$9.1$</td>
<td>$-.2$</td>
</tr>
</tbody>
</table>

Vectors from a recurrent neural network.
Neural N-gram Models

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

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

1 Smoothing
   Neural Networks
   Backoff

2 Implementation Issues
Backoff Smoothing

“in the Ugrasena” was not seen → 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”

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

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

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

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

“the Ugrasena” was not seen $\rightarrow$ 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>Bigrams</th>
<th>Trigrams</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Words</strong></td>
<td><strong>Words</strong></td>
<td><strong>Words</strong></td>
</tr>
<tr>
<td>log p</td>
<td>log b</td>
<td>log p</td>
</tr>
<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>
## Example Language Model

<table>
<thead>
<tr>
<th></th>
<th>Unigrams</th>
<th></th>
<th>Bigrams</th>
<th></th>
<th>Trigrams</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Words</td>
<td>log p</td>
<td>log b</td>
<td>Words</td>
<td>log p</td>
</tr>
<tr>
<td>&lt;s&gt;</td>
<td>−∞</td>
<td>−2.0</td>
<td>−4.1</td>
<td>&lt;s&gt; iran</td>
<td>−3.3</td>
</tr>
<tr>
<td>iran</td>
<td>−4.1</td>
<td>−0.8</td>
<td>−2.5</td>
<td>iran is</td>
<td>−1.7</td>
</tr>
<tr>
<td>is</td>
<td>−2.5</td>
<td>−1.4</td>
<td>−3.3</td>
<td>is one</td>
<td>−2.0</td>
</tr>
<tr>
<td>one</td>
<td>−3.3</td>
<td>−0.9</td>
<td></td>
<td>one of</td>
<td>−1.4</td>
</tr>
<tr>
<td>of</td>
<td>−2.5</td>
<td>−1.1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Query

\[
\log p(\text{is} | <s> \text{ iran}) = -1.1
\]
Example Language Model

<table>
<thead>
<tr>
<th>Unigrams</th>
<th>Words</th>
<th>log p</th>
<th>log b</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;s&gt;</td>
<td>−∞</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>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>
<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>Words</th>
<th>log p</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;s&gt; iran is</td>
<td>−1.1</td>
<td></td>
</tr>
<tr>
<td>iran is one</td>
<td>−2.0</td>
<td></td>
</tr>
<tr>
<td>is one of</td>
<td>−0.3</td>
<td></td>
</tr>
</tbody>
</table>

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

\[
\begin{align*}
\log p(\text{of}) &= -2.5 \\
\log b(\text{is}) &= -1.4 \\
\log b(\text{iran is}) &= +0.4 \\
\end{align*}
\]

\[
\log p(\text{of} | \text{iran is}) = -4.3
\]
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

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

Note: stupid does not sum to 1.

[Brants et al, 2007]
Counting $n$-grams

<s> Australia is one of the few

<table>
<thead>
<tr>
<th>5-gram</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;s&gt; 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 from $n$-gram to count.
stupid($w_n \mid w_{n-1}^1$) = \begin{cases} 
\frac{\text{count}(w_1^n)}{\text{count}(w_{n-1}^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}

\text{count(is one of the) = 12}
stupid($w_n \mid w_{n-1}^n$) = \begin{align*}
\frac{\text{count}(w_{n}^n)}{\text{count}(w_{n-1}^{n-1})} & \quad \text{if } \text{count}(w_{1}^n) > 0 \\
0.4 \text{stupid}(w_n \mid w_{2}^{n-1}) & \quad \text{if } \text{count}(w_{1}^n) = 0
\end{align*}

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

\begin{align*}
\text{count(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
\end{align*}
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: $\sqrt{2^{64}} = 2^{32}$.

$\implies$ 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 __gnucxx::hash_map

$n$-grams

Bucket array

Lookup requires two random memory accesses.
1.5 buckets/entry (so buckets = 6).

- Ideal bucket = hash mod buckets.
- Resolve *bucket* collisions using the next free bucket.

### Bigrams

<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>
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 $[0, |n - \text{grams}|)$
→ Use these as array offsets.

Entries not in the model get assigned offsets
→ Store a fingerprint of each $n$-gram
Minimal Perfect Hash Table

Maps every $n$-gram to a unique integer $[0, |n-grams|)$

→ Use these as array offsets.

Low memory, but potential for false positives
Sorted Array

Sort $n$-grams, perform binary search.

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

---

Introduction

Smoothing

Estimating

Querying

Conclusion
Sorted Array

Sort $n$-grams, perform binary search.

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

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

Entries

- probing
- hash_set
- unordered_set
- interpolation
- binary_search
- set

Introduction
Smoothing
Estimating
Querying
Conclusion
Reverse \( n \)-grams, arrange in a trie.

\[
\begin{align*}
\langle s \rangle & \quad \rightarrow \quad \text{one} & \quad \rightarrow \quad \text{is} \\
& \quad \rightarrow \quad \text{are} & \quad \rightarrow \quad \langle s \rangle \\
\text{is} & \quad \rightarrow \quad \text{Australia} & \quad \rightarrow \quad \text{is} & \quad \rightarrow \quad \text{Australia} \\
\text{one} & \quad \rightarrow \quad \text{are} & \quad \rightarrow \quad \text{are} \\
\text{Australia} & \quad \rightarrow \quad \langle s \rangle
\end{align*}
\]
Quantization: store approximate values
Collapse probability and backoff
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.