N-Gram Language Modelling including Feed-Forward NNs

Kenneth Heafield
University of Edinburgh
History of Language Model History

Kenneth Heafield
University of Edinburgh
Predictive typing

Ty type types typical typing Tyler
$p(\text{type} \mid \text{Predictive}) > p(\text{Tyler} \mid \text{Predictive})$
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]
Heated indoor swimming pool
présidente de la Chambre des représentants

chairwoman of the House 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

\[ p(\text{Moses Compiles}) = ? \]
Sequence Models

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

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

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

Explicit begin and end of sentence.
Sequence Model

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

Where do these probabilities come from?
Probabilities from Text

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

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

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

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

\[
p(\text{Ugrasena} \mid \text{in the}) = \frac{\text{count(\text{in the Ugrasena})}}{\text{count(\text{in the})}} = 0? \\
= \frac{\text{count(\text{the Ugrasena})}}{\text{count(\text{the})}} = 2.07 \cdot 10^{-9}
\]
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 Backoff: maybe “the Ugrasena” was seen?
2 Neural Networks: classifier predicts next word
"in the Ugrasena" was not seen $\rightarrow$ try "the Ugrasena"

$p(Ugrasena \mid \text{in the}) \approx p(Ugrasena \mid \text{the})$
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 → 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 → try “Ugrasena”

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

Backoff \( b \) is a penalty for not matching context.
### Backoff 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>
### Backoff 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**

$$\log p(is \mid <s> \text{ iran}) = -1.1$$
## Backoff 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>log ( p )</strong></td>
<td><strong>log ( b )</strong></td>
</tr>
<tr>
<td>&lt;s&gt;</td>
<td>−∞</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>

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

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

Though long-distance matters:
Grammatical structure
Topical coherence
Words tend to repeat
Cross-sentence dependencies
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</th>
<th>Trigam</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>are one of</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>is one of</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>are two of</td>
<td>3</td>
</tr>
</tbody>
</table>

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

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

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

Save mass for unseen events

Normalize

<table>
<thead>
<tr>
<th>Input 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>

<table>
<thead>
<tr>
<th>Output 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.

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

\[ p(\text{of|one}) = \text{pseudo}(\text{of | one}) + \text{backoff}(\text{one})p(\text{of}) \]
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.

\[ p(\text{of} | \text{one}) = \text{pseudo}(\text{of} | \text{one}) + \text{backoff}(\text{one})p(\text{of}) \]

<table>
<thead>
<tr>
<th>Input n-gram</th>
<th>Input pseudo</th>
<th>Interpolation weight</th>
<th>Output n-gram</th>
<th>Output p</th>
</tr>
</thead>
<tbody>
<tr>
<td>of</td>
<td>0.1</td>
<td>backoff((\epsilon)) = 0.1</td>
<td>of</td>
<td>0.110</td>
</tr>
<tr>
<td>one of</td>
<td>0.2</td>
<td>backoff(\text{one}) = 0.3</td>
<td>one of</td>
<td>0.233</td>
</tr>
<tr>
<td>are one of</td>
<td>0.4</td>
<td>backoff(\text{are one}) = 0.2</td>
<td>are one of</td>
<td>0.447</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?
“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> Australia is one of the few

\begin{tabular}{l|c}
5-gram & Count \\
\hline
<s> Australia is one of & 1 \\
Australia is one of the & 1 \\
is one of the few & 1 \\
\end{tabular}

Hash table?
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 the few</td>
<td>1</td>
</tr>
<tr>
<td>Australia is one of the few</td>
<td>1</td>
</tr>
<tr>
<td>is one of the few</td>
<td>1</td>
</tr>
</tbody>
</table>

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

Text

Hash Table

File
Split and Merge

Text

Hash Table

Sort

File

Merge Sort

Hash Table

Sort

File

Introduction

Smoothing

Kneser-Ney

Implementation

Neural N-grams

Conclusion
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\( (w_n \mid w_{1}^{n-1}) = \begin{cases} \frac{\text{count}(w_1^n)}{\text{count}(w_{1}^{n-1})} & \text{if } \text{count}(w_1^n) > 0 \\ 0.4 \text{stupid}(w_n \mid w_{2}^{n-1}) & \text{if } \text{count}(w_1^n) = 0 \end{cases} \)

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

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

\text{count(\text{is one of the})} = 12
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 } \text{count}(w_1^n) > 0 \\
0.4\text{stupid}(w_n \mid w_{n-1}^n) & \text{if } \text{count}(w_1^n) = 0
\end{cases}

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

\begin{align*}
\text{count(\text{is one of the periwinkle})} &= 0 \times \\
\text{count(\text{one of the periwinkle})} &= 0 \times \\
\text{count(\text{of the periwinkle})} &= 0 \times \\
\text{count(\text{the periwinkle})} &= 3 \checkmark
\end{align*}

\text{count(\text{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.
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: $\sqrt{2^{64}} = 2^{32}$.

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

boost::unordered_map and __gnu_cxx::hash_map

![Bucket array with n-grams](image)
Default Hash Table

boost::unordered_map and __gnu_cxx::hash_map

Bucket array

n-grams

0 1 2 3 4 5

Lookup requires two random memory accesses.
Linear Probing Hash Table

- 1.5 buckets/entry (so buckets = 6).
- Ideal bucket = hash \mod \text{buckets}.
- Resolve \textit{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 $[0, |n - \text{grams}|)$
$
\rightarrow \text{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 $[0, |n - grams|)$

$\rightarrow$ 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. $O(\log n)$ time.

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

```
<\$> → one → is → are
<\$> → is → Australia → <\$>
<\$> → one → is → Australia → <\$>
<\$> → are
```
Saving More RAM

- 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
Problem with Backoff: “in the Ugrasena kingdom”

“Ugrasena” is unseen $\rightarrow$ use unigram to predict “kingdom”

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

One rare word of context breaks conditioning.
More Conditioning

Word class/cluster model

Skip-gram: skip one or more context words

Neural networks
More Conditioning

Word class/cluster model

Skip-gram: skip one or more context words

Neural networks
Neural N-grams (aka Feed-Forward)

Language modeling as classification:

**Input** in the Ugrasena

**Output** kingdom

Predict (classify) the next word given preceding words.
Assign each word a unique row
Query: Concatenate Vectors

\[
\begin{pmatrix}
1 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
1 \\
0 \\
0 \\
1 \\
0 \\
0 \\
0 \\
1 \\
0 \\
0 \\
0 \\
0 \\
0 \\
1 \\
0 \\
0 \\
1 \\
0 \\
0 \\
0 \\
0 \\
0 \\
1 \\
0 \\
0 \\
0 \\
0 \\
1 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\n0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 
\end{pmatrix}
\]

in

the

Ugrasena

Neural Network

\[
\begin{pmatrix}
0.01 \\
0.92 \\
0.04 \\
0.03
\end{pmatrix}
\]
The input has $|\text{vocabulary}| \cdot |\text{history}|$ dimensions.

$\implies$

The input matrix has $|\text{vocabulary}| \cdot |\text{history}| \cdot |\text{hidden}|$ parameters.
Say $|\text{vocabulary}| \approx 1$ billion. About 1 trillion parameters.
Dealing with Vocabulary Size

- **Word vectors**
- Map unpopular words to unknown or part of speech
- Don’t use words: characters or short snippets
Turning Words into Vectors

Vectors from the input matrix
... or your favorite ACL paper.
Use Word Vectors

Steal word representation from another task (language modeling, IR, SVD). Reduce from \(|\text{vocab}|\) one-hot to \(\approx 100\) dimensional dense input.

**Pro**  Other task might have more data

**Con**  Does not jointly learn word representation
Output vocabulary

Output is still \(|\text{vocab}|\)-way classification.

\[ \Rightarrow \] Large output matrix, lots to normalize over.

\[ \Rightarrow \] Slow evaluation, overfitting.
Dealing with Vocabulary Size

- Word vectors
- Map unpopular words to unknown or part of speech
- Don’t use words: characters or short snippets
Translation Modeling

\[ p(\text{bedroom} \mid \text{of the American, de la Chambre des représentants des États-Unis}) \]

[Devin et al 2014]
Conclusion

Language models measure fluency.

Neural networks and backoff are the dominant formalisms.

... But you should probably use recurrent networks.