Efficient Hashing

- **Closed address hashing**
  - Resolve collisions with chains
  - Easier to understand but bigger

- **Open address hashing**
  - Resolve collisions with probe sequences
  - Smaller but easy to mess up

- **Direct-address hashing**
  - No collision resolution
  - Just eject previous entries
  - Not suitable for core LM storage
Integer Encodings

word ids

7  1  15

the  cat  laughed

n-gram  count

233
Bit Packing

Got 3 numbers under $2^{20}$ to store?

<table>
<thead>
<tr>
<th>Number</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>0...0011</td>
</tr>
<tr>
<td>1</td>
<td>0...00001</td>
</tr>
<tr>
<td>15</td>
<td>0...0111</td>
</tr>
</tbody>
</table>

Fits in a primitive 64-bit long
Integer Encodings

n-gram encoding

15176595 = 20 bits 20 bits 20 bits

the cat laughed

n-gram count

233
c(the) = 23135851162 < 2^{35}

35 bits to represent integers between 0 and 2^{35}
# unique counts = 770000 < 2^{20}

20 bits to represent ranks of all counts

n-gram encoding 60 bits 15176595 20 bits 3 rank

<table>
<thead>
<tr>
<th>rank</th>
<th>freq</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>51</td>
</tr>
<tr>
<td>3</td>
<td>233</td>
</tr>
</tbody>
</table>
So Far

**Word indexer**

<table>
<thead>
<tr>
<th>word</th>
<th>id</th>
</tr>
</thead>
<tbody>
<tr>
<td>cat</td>
<td>0</td>
</tr>
<tr>
<td>the</td>
<td>1</td>
</tr>
<tr>
<td>was</td>
<td>2</td>
</tr>
<tr>
<td>ran</td>
<td>3</td>
</tr>
</tbody>
</table>

**N-gram encoding scheme**

- **unigram**: $f(id) = id$
- **bigram**: $f(id_1, id_2) = ?$
- **trigram**: $f(id_1, id_2, id_3) = ?$

**Rank lookup**

<table>
<thead>
<tr>
<th>rank</th>
<th>freq</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>51</td>
</tr>
<tr>
<td>3</td>
<td>233</td>
</tr>
</tbody>
</table>

**Count DB**

<table>
<thead>
<tr>
<th></th>
<th>unigram</th>
<th>bigram</th>
<th>trigram</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>16078820 0381</td>
<td>16078820 0381</td>
<td>16078820 0381</td>
</tr>
<tr>
<td></td>
<td>15176595 0051</td>
<td>15176595 0051</td>
<td>15176595 0051</td>
</tr>
<tr>
<td></td>
<td>15176583 0076</td>
<td>15176583 0076</td>
<td>15176583 0076</td>
</tr>
<tr>
<td></td>
<td>___</td>
<td>___</td>
<td>___</td>
</tr>
<tr>
<td></td>
<td>16576628 0021</td>
<td>16576628 0021</td>
<td>16576628 0021</td>
</tr>
<tr>
<td></td>
<td>___</td>
<td>___</td>
<td>___</td>
</tr>
<tr>
<td></td>
<td>15176600 0018</td>
<td>15176600 0018</td>
<td>15176600 0018</td>
</tr>
<tr>
<td></td>
<td>16089320 0171</td>
<td>16089320 0171</td>
<td>16089320 0171</td>
</tr>
<tr>
<td></td>
<td>15176583 0039</td>
<td>15176583 0039</td>
<td>15176583 0039</td>
</tr>
<tr>
<td></td>
<td>14980420 0030</td>
<td>14980420 0030</td>
<td>14980420 0030</td>
</tr>
<tr>
<td></td>
<td>___</td>
<td>___</td>
<td>___</td>
</tr>
<tr>
<td></td>
<td>15020330 0482</td>
<td>15020330 0482</td>
<td>15020330 0482</td>
</tr>
</tbody>
</table>
# Hashing vs Sorting

<table>
<thead>
<tr>
<th>Sorting</th>
<th>Hashing</th>
</tr>
</thead>
<tbody>
<tr>
<td>(c)</td>
<td>(c)</td>
</tr>
<tr>
<td>(15176583)</td>
<td>(16078820)</td>
</tr>
<tr>
<td>(15176595)</td>
<td>(15176595)</td>
</tr>
<tr>
<td>(15176600)</td>
<td>(15176583)</td>
</tr>
<tr>
<td>(16078820)</td>
<td>(16576628)</td>
</tr>
<tr>
<td>(16089320)</td>
<td>(16089320)</td>
</tr>
<tr>
<td>(16576628)</td>
<td>(15176600)</td>
</tr>
<tr>
<td>(16980420)</td>
<td>(16089320)</td>
</tr>
<tr>
<td>(17020330)</td>
<td>(15176583)</td>
</tr>
<tr>
<td>(17176583)</td>
<td>(14980420)</td>
</tr>
<tr>
<td>(0076)</td>
<td>(0381)</td>
</tr>
<tr>
<td>(0051)</td>
<td>(0051)</td>
</tr>
<tr>
<td>(0018)</td>
<td>(0076)</td>
</tr>
<tr>
<td>(0381)</td>
<td>(0021)</td>
</tr>
<tr>
<td>(0171)</td>
<td>(0171)</td>
</tr>
<tr>
<td>(0021)</td>
<td>(0018)</td>
</tr>
<tr>
<td>(0030)</td>
<td>(0039)</td>
</tr>
<tr>
<td>(0482)</td>
<td>(0030)</td>
</tr>
<tr>
<td>(039)</td>
<td>(0482)</td>
</tr>
</tbody>
</table>
Maximum Entropy Models
Improving on N-Grams?

- N-grams don’t combine multiple sources of evidence well

\[ P(\text{construction} \mid \text{After the demolition was completed, the}) \]

- Here:
  - “the” gives syntactic constraint
  - “demolition” gives semantic constraint
  - Unlikely the interaction between these two has been densely observed in this specific n-gram

- We’d like a model that can be more statistically efficient
Some Definitions

INPUTS

\(X_i\)

CANDIDATE SET

\(\mathcal{Y}(x)\)

\{door, table, \ldots\}

CANDIDATES

\(y\)

TRUE OUTPUTS

\(y^*_i\)

door

FEATURE VECTORS

\(f(x, y)\) [0 0 1 0 0 0 1 0 0 0 0 0]

- \(x_{-1} = \text{“the”} \land y = \text{“door”}\)
- \(x_{-1} = \text{“the”} \land y = \text{“table”}\)
- \(\text{“close” in } x \land y = \text{“door”}\)
- \(y \text{ occurs in } x\)
More Features, Less Interaction

\[ x = \text{closing the} \quad \_\_\_\_ , \quad y = \text{doors} \]

- **N-Grams** \[ x_{-1} = \text{“the”} \land y = \text{“doors”} \]
- **Skips** \[ x_{-2} = \text{“closing”} \land y = \text{“doors”} \]
- **Lemmas** \[ x_{-2} = \text{“close”} \land y = \text{“door”} \]
- **Caching** \[ y \text{ occurs in } x \]
## Data: Feature Impact

<table>
<thead>
<tr>
<th>Features</th>
<th>Train Perplexity</th>
<th>Test Perplexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 gram indicators</td>
<td>241</td>
<td>350</td>
</tr>
<tr>
<td>1-3 grams</td>
<td>126</td>
<td>172</td>
</tr>
<tr>
<td>1-3 grams + skips</td>
<td>101</td>
<td>164</td>
</tr>
</tbody>
</table>
Exponential Form

- Weights $w$  Features $f(x, y)$

- Linear score $w^\top f(x, y)$

- Unnormalized probability
  $$P(y|x, w) \propto \exp(w^\top f(x, y))$$

- Probability
  $$P(y|x, w) = \frac{\exp(w^\top f(x, y))}{\sum_{y'} \exp(w^\top f(x, y'))}$$
Likelihood Objective

- Model form:

\[
P(y|x, w) = \frac{\exp(w^\top f(x, y))}{\sum_{y'} \exp(w^\top f(x, y'))}
\]

- Log-likelihood of training data

\[
L(w) = \log \prod_i P(y_i^*|x_i, w) = \sum_i \log \left( \frac{\exp(w^\top f(x_i, y_i^*))}{\sum_{y'} \exp(w^\top f(x_i, y'))} \right)
\]

\[
= \sum_i \left( w^\top f(x_i, y_i^*) - \log \sum_{y'} \exp(w^\top f(x_i, y')) \right)
\]
Training
History of Training

- **1990’s**: Specialized methods (e.g. iterative scaling)
- **2000’s**: General-purpose methods (e.g. conjugate gradient)
- **2010’s**: Online methods (e.g. stochastic gradient)
What Does LL Look Like?

- **Example**
  - Data: xxxy
  - Two outcomes, x and y
  - One indicator for each
  - Likelihood

\[
\log \left( \left( \frac{e^x}{e^x + e^y} \right)^3 \times \frac{e^y}{e^x + e^y} \right)
\]
The maxent objective is an unconstrained convex problem

\[ L(w) \]

\[ \nabla L(w) = 0 \]

One optimal value*, gradients point the way
Gradients

\[ L(w) = \sum_i \left( w^T f(x_i, y_i^*) - \log \sum_y \exp(w^T f(x_i, y)) \right) \]

\[ \frac{\partial L(w)}{\partial w} = \sum_i \left( f(x_i, y_i^*) - \sum_y P(y|x_i)f(x_i, y) \right) \]

Count of features under target labels

Expected count of features under model predicted label distribution
Gradient Ascent

- The maxent objective is an unconstrained optimization problem

\[ L(w) \]

Gradient Ascent

- Basic idea: move uphill from current guess
- Gradient ascent / descent follows the gradient incrementally
- At local optimum, derivative vector is zero
- Will converge if step sizes are small enough, but not efficient
- All we need is to be able to evaluate the function and its derivative
(Quasi)-Newton Methods

- 2\textsuperscript{nd}-Order methods: repeatedly create a quadratic approximation and solve it

\[ L(w) \]

\[ L(w_0) + \nabla L(w)^\top (w - w_0) + (w - w_0)^\top \nabla^2 L(w)(w - w_0) \]

- E.g. LBFGS, which tracks derivative to approximate (inverse) Hessian
Regularization
Regularization Methods

- Early stopping
- L2: $L(w) - |w|^2$
- L1: $L(w) - |w|$
Regularization Effects

- Early stopping: don’t do this
- L2: weights stay small but non-zero
- L1: many weights driven to zero
  - Good for sparsity
  - Usually bad for accuracy for NLP
Scaling
Why is Scaling Hard?

$L(w) = \sum_i \left( w^\top f(x_i, y^*_i) - \log \sum_y \exp(w^\top f(x_i, y)) \right)$

- Big normalization terms
- Lots of data points
Hierarchical Prediction

- Hierarchical prediction / softmax [Mikolov et al 2013]

- Noise-Contrastive Estimation [Mnih, 2013]

- Self-Normalization [Devlin, 2014]
Stochastic Gradient

- View the gradient as an average over data points

\[
\frac{\partial L(w)}{\partial w} = \frac{1}{N} \sum_i \left( f(x_i, y_i^*) - \sum_y P(y|x_i)f(x_i, y) \right)
\]

- Stochastic gradient: take a step each example (or mini-batch)

\[
\frac{\partial L(w)}{\partial w} \approx \frac{1}{1} \left( f(x_i, y_i^*) - \sum_y P(y|x_i)f(x_i, y) \right)
\]

- Substantial improvements exist, e.g. AdaGrad (Duchi, 11)
Other Methods
Neural Net LMs

\[ i-th \ output = P(w_t = i \mid context) \]

Image: (Bengio et al, 03)
Neural vs Maxent

- Maxent LM

\[ P(y|x, w) \propto \exp(w^\top f(x, y)) \]

- Neural Net LM

\[ P(y|x, w) \propto \exp(B\sigma(Af(x))) \]

\( \sigma \) nonlinear, e.g. tanh
Neural Net LMs

\[ P(y|w, x) \propto e^{Bh} \]

\[ h = \sigma \left( \sum_i A_i v_i \right) \]

\[ x_{-2} = \text{closing} \]
\[ x_{-1} = \text{the} \]
Maximum Entropy LMs

- Want a model over completions $y$ given a context $x$:

$$P(y|x) = P(\text{close the door} | \text{close the} )$$

- Want to characterize the important aspects of $y = (v,x)$ using a feature function $f$

- $F$ might include
  - Indicator of $v$ (unigram)
  - Indicator of $v$, previous word (bigram)
  - Indicator whether $v$ occurs in $x$ (cache)
  - Indicator of $v$ and each non-adjacent previous word
  - …