Algorithms for NLP

Language Modeling III

Taylor Berg-Kirkpatrick – CMU

Slides: Dan Klein – UC Berkeley
Tries

this -1.2

is -9.7

that -3.5

is -4.7

a -4.5

the -3.7

4-gram -8.4

4-gram -6.8

4-gram -3.2

4-gram -2.0

[Hsu and Glass 2008]
Context Encodings

40-bits  24-bits

548029639  678  431

4-gram  -8.7

Google N-grams
- 10.5 bytes/n-gram
- 37 GB total

[Many details from Pauls and Klein, 2011]
# Context Encodings

<table>
<thead>
<tr>
<th>1-grams</th>
<th>2-grams</th>
<th>3-grams</th>
</tr>
</thead>
<tbody>
<tr>
<td>( w )</td>
<td>( val )</td>
<td>( c )</td>
</tr>
<tr>
<td>675</td>
<td>0127</td>
<td>15176582</td>
</tr>
<tr>
<td>676</td>
<td>9008</td>
<td>15176583</td>
</tr>
<tr>
<td>677</td>
<td>0137</td>
<td>15176584</td>
</tr>
<tr>
<td>678</td>
<td>0090</td>
<td>15176585</td>
</tr>
<tr>
<td>679</td>
<td>1192</td>
<td>15176586</td>
</tr>
<tr>
<td>680</td>
<td>0050</td>
<td>15176587</td>
</tr>
<tr>
<td>681</td>
<td>0040</td>
<td>15176588</td>
</tr>
<tr>
<td>682</td>
<td>0201</td>
<td>15176589</td>
</tr>
<tr>
<td>683</td>
<td>3010</td>
<td>15176590</td>
</tr>
</tbody>
</table>

- **20 bits**
- **64 bits**
- **20 bits**

- **“this”**
- **“a”**
- **“the”**
- **“was”**
- **“is”**
- **“a”**
- **“the”**

- **“is”**
- **“a”**
- **“the”**
Compression
### Idea: Differential Compression

| $c$        | $w$  | $\text{val}$ | $\Delta c$ | $\Delta w$ | $\text{val}$ | $|\Delta w|$ | $|\Delta c|$ | $|\text{val}|$ |
|------------|------|--------------|------------|------------|--------------|-------------|-------------|-------------|
| 15176585   | 678  | 3            |            |            |              |             |             |             |
| 15176587   | 678  | 2            | +2         | +0         | 2            | 3           | 2           | 3           |
| 15176593   | 678  | 1            | +6         | +0         | 1            | 3           | 2           | 3           |
| 15176613   | 678  | 8            | +40        | +0         | 8            | 9           | 2           | 6           |
| 15179801   | 678  | 1            | +188       | +0         | 1            | 12          | 2           | 3           |
| 15176585   | 680  | 298          |            | +2         | 298          | 36          | 4           | 15          |
| 15176589   | 680  | 1            | +4         | +0         | 1            | 6           | 2           | 3           |

| $c$        | $w$  | $\text{val}$ | $\Delta c$ | $\Delta w$ | $\text{val}$ | $|\Delta w|$ | $|\Delta c|$ | $|\text{val}|$ |
|------------|------|--------------|------------|------------|--------------|-------------|-------------|-------------|
| 15176585   | 678  | 3            |            |            |              |             |             |             |
| 563097887  | 956  | 3            | +2         | +0         | 2            | +6          | +0          | +40         | +2          | 8           | ...         |
Variable Length Encodings

Encoding “9”

<table>
<thead>
<tr>
<th>Length in Unary</th>
<th>Number in Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>1001</td>
</tr>
</tbody>
</table>

Google N-grams
- 2.9 bytes/n-gram
- 10 GB total

[Elias, 75]
Speed-Ups
## Context Encodings

<table>
<thead>
<tr>
<th>1-grams</th>
<th>2-grams</th>
<th>3-grams</th>
</tr>
</thead>
<tbody>
<tr>
<td>w</td>
<td>val</td>
<td>c</td>
</tr>
<tr>
<td>675</td>
<td>0127</td>
<td>&quot;this&quot;</td>
</tr>
<tr>
<td>676</td>
<td>9008</td>
<td></td>
</tr>
<tr>
<td>677</td>
<td>0137</td>
<td></td>
</tr>
<tr>
<td>678</td>
<td>0090</td>
<td>&quot;a&quot;</td>
</tr>
<tr>
<td>679</td>
<td>1192</td>
<td></td>
</tr>
<tr>
<td>680</td>
<td>0050</td>
<td>&quot;the&quot;</td>
</tr>
<tr>
<td>681</td>
<td>0040</td>
<td></td>
</tr>
<tr>
<td>682</td>
<td>0201</td>
<td>&quot;is&quot;</td>
</tr>
<tr>
<td>683</td>
<td>3010</td>
<td>&quot;was&quot;</td>
</tr>
<tr>
<td>20 bits</td>
<td>20 bits</td>
<td>64 bits</td>
</tr>
</tbody>
</table>

- 20 bits for each word
- 64 bits for each character
- 20 bits for each value

**Legend:**
- "this"
- "a"
- "the"
- "was"
- "is"
Naïve N-Gram Lookup

This is a 4-gram

\[ p(0121 \ 0374 \ 0045 \ 4820) = -8.7 \]
Rolling Queries

this is + a 4-gram
12438010 0045 4820

<table>
<thead>
<tr>
<th>c</th>
<th>w</th>
<th>val</th>
<th>suffix</th>
</tr>
</thead>
<tbody>
<tr>
<td>15176583</td>
<td>682</td>
<td>0065</td>
<td>00000480</td>
</tr>
<tr>
<td>15176595</td>
<td>682</td>
<td>0808</td>
<td>00000675</td>
</tr>
<tr>
<td>15176600</td>
<td>682</td>
<td>0012</td>
<td>00000802</td>
</tr>
<tr>
<td>16078820</td>
<td>682</td>
<td>0400</td>
<td>00001321</td>
</tr>
</tbody>
</table>

12438010 0045 → LM [val] -7.8

15176583 4820 → LM [val] -5.4
Idea: Fast Caching

<table>
<thead>
<tr>
<th>n-gram</th>
<th>probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>124 80 42 1243</td>
<td>-7.034</td>
</tr>
<tr>
<td>37 2435 243 21</td>
<td>-2.394</td>
</tr>
<tr>
<td>804 42 4298 43</td>
<td>-8.008</td>
</tr>
</tbody>
</table>

\[
\text{hash}(124 \ 80 \ 42 \ 1243) = 0
\]

\[
\text{hash}(1423 \ 43 \ 42 \ 400) = 1
\]

LM can be more than 10x faster w/ direct-address caching
Approximate LMs

- Simplest option: hash-and-hope
  - Array of size $K \sim N$
  - (optional) store hash of keys
  - Store values in direct-address
  - Collisions: store the max
  - What kind of errors can there be?

- More complex options, like bloom filters (originally for membership, but see Talbot and Osborne 07), perfect hashing, etc
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

$X_i$

$X_i$ close the ____

CANDIDATE SET

$\mathcal{Y}(x)$

$\mathcal{Y}(x) \{\text{door, table, ...}\}$

CANDIDATES

$y$

$y$ table

TRUE OUTPUTS

$y^*$

$y^*$ door

FEATURE VECTORS

$f(x, y) \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$

$x_{-1} = \text{"the" \& y = "door"}$

"close" in $x$ \& $y = \text{"door"}$

$x_{-1} = \text{"the" \& y = "table"}$

"y occurs in $x$"
More Features, Less Interaction

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

- **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** \( w^T f(x, y) \)

- **Unnormalized probability**

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

- **Probability**

\[
P(y|x, w) = \frac{\exp(w^T f(x, y))}{\sum_{y'} \exp(w^T f(x, y'))}
\]
### 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

\[
\log \left( \left( \frac{e^x}{e^x + e^y} \right)^3 \times \frac{e^y}{e^x + e^y} \right)
\]
Convex Optimization

- The maxent objective is an unconstrained convex problem

\[ L(w) \]

- 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
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

- $L_2$: $L(w) - |w|^2$

- $L_1$: $L(w) - |w|$
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?

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

- 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

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

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

- **Maxent LM**

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

- **Neural Net LM**

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

\( \sigma \) nonlinear, e.g. tanh
Mixed Interpolation

- But can’t we just interpolate:
  - $P(w | \text{most recent words})$
  - $P(w | \text{skip contexts})$
  - $P(w | \text{caching})$
  - ...

- Yes, and people do (well, did)
  - But additive combination tends to flatten distributions, not zero out candidates
- **Decision trees?**
  - Good for non-linear decision problems
  - Random forests can improve further [Xu and Jelinek, 2004]
  - Paths to leaves basically learn conjunctions
  - General contrast between DTs and linear models