NLP

- NLP is a large area
- Primary work is current in Machine Learning
- Book touches on small fraction of interesting ongoing work
- We will stick to the book

Language Models

- Formal Language
  - Programming language or logic language
  - Unambiguous
  - Defined by grammar/semantics
  - Does not change

- Natural Language
  - English, Spanish, Chinese
  - Ambiguous
  - No 100% agreed upon grammar/semantics
  - Constantly change
Language Examples (many via wikipedia)

- I saw the man with the binoculars.
- The complex houses married and single soldiers and their families.
- The horse raced past the barn fell.
- Look at the dog with one eye.

- Colorless green ideas sleep furiously

N-gram Character models

- Suppose we are interested in sequences of characters:
  - $P(c_{1:N})$
  - Probability of the sequence of N characters
- Special case: bigram, trigram
  - N-gram is N-1 Markov
  - Model as depending on previous N-1 characters
  - $P(c_i | c_{1:i-1}) = P(c_i | c_{i-2:i-1})$
  - $P(c_{1:N}) = \prod_{i=1}^{N} P(c_i | c_{1:i-1}) = \prod_{i=1}^{N} P(c_i | c_{i-2:i-1})$

N-gram Character models

- Training text is a corpus
  - If we have 100 characters, $100^3$ trigrams = 1 million
  - How many english trigrams without spaces?

Trigrams

<table>
<thead>
<tr>
<th>Common</th>
<th>Uncommon</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>the</td>
<td>82,103,550,112</td>
<td>jwq 10,340</td>
</tr>
<tr>
<td>ing</td>
<td>43,727,954,927</td>
<td>jqy 8,871</td>
</tr>
<tr>
<td>and</td>
<td>43,452,082,914</td>
<td>zqy 8,474</td>
</tr>
<tr>
<td>ion</td>
<td>39,907,843,075</td>
<td>jzq 7,180</td>
</tr>
<tr>
<td>tio</td>
<td>32,705,432,538</td>
<td>zgq 6,254</td>
</tr>
</tbody>
</table>

http://norvig.com/ngrams/count_3l.txt

What is the most common trigram in your first name?
Task: Language prediction

- How can we predict the language used in text?
  - $\ell^* = \arg \max_{\ell} P(\ell | c_{1:N})$
  - $= \arg \max_{\ell} P(\ell) P(c_{1:N} | \ell)$
  - $= \arg \max_{\ell} P(\ell) \prod_{i=1}^{N} P(c_i | c_{i-1:i-1} \ell)$
  - $(c_{i-2})$

Caveats

- What if we don’t see a particular $n$-gram in our data?
  - Simple: Give every $n$-gram a count of at least 1
  - Complex: Smooth with lower $n$-gram models

$P(c_i | c_{i-2:i-1}) = \lambda_3 P(c_i | c_{i-2:i-1}) + \lambda_2 P(c_i | c_{i-1}) + \lambda_1 P(c_i)$

N-grams on words

- Can not only be used for classification, but for reconstruction of “probable” sentences
  - Unigram: logical are as are confusion a may right tries agent goal the was
  - Bigram: systems are very similar computational approach would be represented
  - Trigram: planning and scheduling are integrated the success of naive bayes model is
  - Take your initials and find the most common word

http://norvig.com/ngrams/count_3l.txt

Text classification

- Categorize e-mails: spam or not?
  - Bayes approach:
    - Treat spam messages and ham (not-spam) messages as two different languages
    - Then, classify which language the message is
Text classification

• Categorize e-mails: spam or not?
• Machine Learning Approach (1):
  • Use a “bag of words” model
  • One binary feature for each word
  • Train classifier to predict

Text classification

• Categorize e-mails: spam or not?
• Machine Learning Approach (2):
  • Add many features based on the e-mail
    • Sent date/time
    • Embedded image check
    • URL check
    • Sender in mailbox

Information Retrieval (IR)

• Given a query, find related documents
  • Library search (relevant books)
  • Google search (relevant web pages)
• Uses:
  • Corpus of documents
  • Queries (and query language)
  • Result set
  • (Result presentation)

IR evaluation

• Precision:
  • % of relevant documents in the query
• Recall:
  • % of possible relevant documents returned

• What if we try to maximize precision?
• What if we try to maximize recall?
**PageRank (Google)**

\[
PR(p) = \frac{1 - d}{N} + d \sum_i \frac{PR(in_i)}{C(in_i)}
\]

- \(N\) = # of documents in corpus
- \(in_i\) = pages that link to \(p\)
- \(C(in_i)\) = count of out links on page \(in_i\)
- \(d\) = dampening factor

**Information Extraction**

- Given a corpus how do we extract relevant features?
- Easy idea:
  - Finite State Machines
- We won’t cover more complex ideas

**Finite-State Machine (Automata) Extraction**

- Assume structured text about a single topic
- Task: extract prices from amazon
- Can look up product by ISBN:
  - \(http://www.amazon.com/dp/0136042597/\)

**Amazon price extraction**

```
curl -A "Mozilla/4.0" "http://www.amazon.com/dp/0136042597/
```
Amazon price extraction

curl -A "Mozilla/4.0" "http://www.amazon.com/dp/0136042597/"
| grep '^\$'


$45
$178
$132
$19
$45
$158
$132
$19
$158
$137
$178
$40
$92
Amazon price extraction

curl -A "Mozilla/4.0" "http://www.amazon.com/dp/0136042597/" | grep -E '\$[0-9]+[.]*[0-9]*\$'

<table>
<thead>
<tr>
<th>rentPrice</th>
<th>rentListPrice</th>
<th>rentListPriceSavings</th>
</tr>
</thead>
<tbody>
<tr>
<td>$45.25</td>
<td>$178.20</td>
<td>$132.95 (75%)</td>
</tr>
</tbody>
</table>

Finite State Machine Example (extract comics)

set x = 9chickweedlane
curl -A "Mozilla/4.0" http://www.gocomics.com/$x/2013/10/13 | grep img

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
<!-- General Meta Tags -->
<meta http-equiv="content-type" content="text/html;charset=UTF-8"/>
<!-- Facebook Open Graph Tags -->
<meta property="og:type" content="gocomics:comic" />
<meta property="og:site_name" content="GoComics" />
<meta property="og:description" content="One of the..."/>
</head>
</html>

 Finite State Machine Example (extract comics)

set x = 9chickweedlane
curl -A "Mozilla/4.0" http://www.gocomics.com/$x/2013/10/13 | grep "strip"

<p class="feature_item" data-id="1156584"><a href="#mutable_1156584" class="photo"><img alt="9 Chickweed Lane" class="strip" src="http://assets.amuniversal.com/7f2ad120bcf80131556d005056a9545d" width="600" /></a>
</p>
Finite State Machine Example (extract comics)

```
set x = 9chickweedlane
curl -A "Mozilla/4.0" http://www.gocomics.com/$x/2013/10/13 |
grep img | grep "strip" | cut -f 10 -d "" |

http://assets.amuniversal.com/7f2ad120b80131556d005056a9545d
```

Finite State Machine Example (extract comics)

```
set x = 9chickweedlane
curl `curl -A "Mozilla/4.0" http://www.gocomics.com/$x/ |
2013/10/13 | grep img | grep "strip" | cut -f 10 -d "" |
-o $x.jpeg`
```