Introduction to Artificial Intelligence
COMP 3501 / COMP 4704-4
Lecture 17: Natural Language Processing

Prof. Nathan Sturtevant

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

Language Models

- 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

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

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 -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/"
```

```
<!doctype html><!--[if lt IE 7]><html class="a-no-js a-lt-ie10
a-lt-ie9 a-lt-ie8 a-lt-ie7 a-ie6" data-19ax5a9jf="dingo"><!
[endif]--><!--[if IE 7]><html class="a-no-js a-lt-ie10 a-lt-
ie9 a-lt-ie8 a-ie7" data-19ax5a9jf="dingo"><![endif]--><!--[if
IE 8]><html class="a-no-js a-lt-ie10 a-lt-ie9 a-ie8"
data-19ax5a9jf="dingo"><![endif]--><!--[if IE 9]><html
class="a-no-js a-lt-ie10 a-ie9" data-19ax5a9jf="dingo"><!
[endif]--><!--[if !IE]><!--><html class="a-no-js"
data-19ax5a9jf="dingo"><!--<![endif]-->
<head>
<script type="text/javascript">var ue_t0=ue_t0||+new Date();</
script>
<script type='text/javascript'>
var ue_id='0QWEBGH576D1A0D3JJ2Y',
ue_csm = window
```

Amazon price extraction

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

```
ue_fpf='//fls-na.amazon.com/1/batch/1/OP/ATVPDKIKX0DER:
180-2406584-1820662:0QWEBGH576D1A0D3JJ2Y$uedata=s:',
ue_csm.ue_hob=ue_csm.ue_hob||+new Date();(function(i,o)
{i.ueinit=(i.ueinit||0)+1;var d={t0:o.aPageStart||
i.ue_t0,id:i.ue_id,url:i.ue_url,rid:i.ue_id,a:"",b:"",h:{},r:
{ld:0,oe:0,ul:0},s:1,t:{},sc:{},iel:[],ielf:[],fc_idx:{},viz:
[],v:
33,d:i.ue&&i.ue.d,log:i.ue&&i.ue.log,lr:i.ue&&i.ue.lr,ulh:
[]},p=i.ue_fpf?1:0,m="beforeunload",e="undefined",h;function
c(q){return q&&q.replace&&q.replace(/^\s+|\s+$/g,"")}
d.oid=c(d.id);d.lid=c(d.id);i.ue=d;i.ue._t0=i.ue.t0;function
j(s){if(!i.ue_fpf||!o.encodeURIComponent||!s){return}var q=new
Image(),r=""+i.ue_fpf+o.encodeURIComponent(s)+":"+(+new
Date()-i.ue_t0);i.ue.iel.push(q);q.src=r}i.ue.tagC=function()
{var q=[];return function(r){if(r){q.push(r)}return
```

Amazon price extraction

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

```
$45
$178
$132
$19
$45
$158
$132
$45
$132
$19
$158
$137
$178
$40
$92
```

Amazon price extraction

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

```
$45.25
$178.20
$132.95
$19.98
$45.25
$158.22
$132.95
$45.25
$132.95
$19.98
$158.22
$137.21
$178.20
$40.99
$92.06
```

Amazon price extraction

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

```
$45.25
$178.20
$132.95
$19.98
$45.25
$158.22
$132.95
$45.25
$132.95
$19.98
$158.22
$137.21
$178.20
$40.99
$92.06
```

Amazon price extraction

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

```
                        <span id="rentPrice" class="a-size-
base a-color-base a-text-normal offer-price">$45.25</span>
                        <span class="a-color-secondary">List
Price:</span> <span id="rentListPrice" class="a-color-
secondary a-text-strike">$178.20</span>
                        <span class="a-size-base a-color-
secondary">Save: </span><span id="rentListPriceSavings"
class="a-color-secondary">$132.95(75%) </span>
            A.$("#rentPrice").html("$19.98 - $45.25");
            A.$("#rentListPriceSavings").text("$158.22 -
$132.95 (89% - 75%)");
            A.$("#rentPrice").html("$45.25");
            A.$("#rentListPriceSavings").text("$132.95" +
" (75%)");
            A.$("#rentPrice").html("$19.98");
```

Finite State Machine Example* (extract comics)

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

```
<!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
```

Finite State Machine Example (extract comics)

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

```
  <a href="/" id="logo"><img alt="GoComics.com Logo"
height="90" src="http://assets.gocomics.com/images/logo-
uclick-gocomics-stacked.png?1400528009" title="GoComics"
width="221" /></a>
        <div class="banner"><a href="http://
pibpress.blogspot.com"><img alt="New-topper" src="http://
images.gocomics.com/images/gocomics/flare/9-chickweed-lane/
new-topper.jpg" /></a></div>
            <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" /><span
class='zoom_link'></span></a><div id='mutable_1156584'
style='display: none;'><img
alt="7fce3ee0bcf80131556d005056a9545d" class="strip"
```

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"'
```

```
            <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" /><span
class='zoom_link'></span></a><div id='mutable_1156584'
style='display: none;'><img
alt="7fce3ee0bcf80131556d005056a9545d" class="strip"
src="http://assets.amuniversal.com/
7fce3ee0bcf80131556d005056a9545d" /></div></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/7f2ad120bcf80131556d005056a9545d

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.jpg
```