Adminstrivia

• Homework 4 graded
• Homework 5 out today
  – Due Saturday night by midnight
  – *(Gives me Sunday to grade it)*
Homework 5

• Write a regex over Part-of-Speech (POS) tags that identifies (as best you can) the basic Noun Phrases (NPs) in text.

• Example (from the UIUC demo, see next slide):
  – [Helicopters] will patrol [the temporary no-fly zone] around [New Jersey's MetLife Stadium] [Sunday], with [F-16s] based in [Atlantic City] ready to be scrambled if [an unauthorized aircraft] does enter [the restricted airspace].
  – Down below, [bomb-sniffing dogs] will patrol [the trains] and [buses] that are expected to take approximately [30,000] of [the 80,000-plus spectators] to [Sunday's Super Bowl] between [the Denver Broncos and Seattle Seahawks].
  – [The Transportation Security Administration] said it has added about [two dozen dogs] to monitor [passengers] coming in and out of [the airport] around [the Super Bowl].
  – On [Saturday], [TSA agents] demonstrated how [the dogs] can sniff out [many different types] of [explosives]. Once they do, [they]’re trained to sit rather than attack, so as not to raise [suspicion] or create [a panic].
  – [TSA spokeswoman Lisa Farbstein] said [the dogs] undergo [12 weeks] of [training], which costs about [$200,000], factoring in [food], [vehicles] and [salaries] for [trainers].
  – [Dogs] have been used in [cargo areas] for [some time], but have just been introduced recently in [passenger areas] at [Newark] and [JFK airports]. [JFK] has [one dog] and [Newark] has [a handful], [Farbstein] said.

for this exercise, basic = the patterns identified in this slide
Homework 5

Part of Speech Tagging Demo

737,457 views

If you wish to cite this work, please cite this publication.

Helicopters will patrol the temporary no-fly zone around New Jersey's MetLife Stadium Sunday, with F-16s based in Atlantic City ready to be scrambled if an unauthorized aircraft does enter the restricted airspace.

Down below, bomb-sniffing dogs will patrol the trains and buses that are expected to take approximately 30,000 of the 80,000-plus spectators to Sunday's Super Bowl between the Denver Broncos and Seattle Seahawks.

The Transportation Security Administration said it has added about two dozen dogs to monitor passengers and luggage at Newark Liberty International Airport.
We will use an online Part-of-Speech (POS) tagger to pre-process the raw text.

http://cogcomp.cs.illinois.edu/page/demo_view/POS

Example:

- **NNPS Helicopters** MD will **NN** patrol **DT** the **JJ** temporary **JJ** no-fly **NN** zone **IN** around **NNP** New **NNP** Jersey **POS** 's **NNP** MetLife **NNP** Stadium **NNP** Sunday **NN** with **IN** **NNP** F-16s **VBN** based **IN** in **NNP** Atlantic **NNP** City **JJ** ready **TO** to **VB** be **VBN** scrambled **IN** if **DT** an **JJ** unauthorized **NN** aircraft **VBZ** does **VB** enter **DT** the **VBN** restricted **NN** airspace ..

**IN** Down **IN** below **, ,** JJ bomb-sniffing **NNS** dogs **MD** will **NN** patrol **DT** the **NNS** trains **CC** and **NNS** buses **WDT** that **VBP** are **VBN** expected **TO** to **VB** take **RB** approximately **CD** 30,000 **IN** of **DT** the **JJ** 80,000-plus **NNS** spectators **TO** to **NNP** Sunday **POS** 's **NNP** Super **NNP** Bowl **IN** between **DT** the **NNP** Denver **NNP** Broncos **CC** and **NNP** Seattle **NNP** Seahawks ..

**DT** The **NNP** Transportation **NNP** Security **NNP** Administration **VBD** said **PRP** it **VBZ** has **VBN** added **IN** about **CD** two **NN** dozen **NNS** dogs **TO** to **VB** monitor **NNS** passengers **VBG** coming **RP** in **CC** and **RP** out **IN** of **DT** the **NN** airport **IN** around **DT** the **NNP** Super **NNP** Bowl ..

**IN** On **NNP** Saturday **, ,** **NNP** TSA **NNP** agents **VBD** demonstrated **WBR** how **DT** the **NNS** dogs **MD** can **VB** sniff **RB** out **JJ** many **JJ** different **NNS** types **IN** of **NNS** explosives .. **RB** Once **PRP** they **VBP** do **, ,** **PRP** they **VBP** 're **VBN** trained **TO** to **VB** sit **RB** rather **IN** than **NN** attack **, ,** **RB** so **RB** as **RB** not **TO** to **VB** raise **NN** suspicion **CC** or **VB** create **DT** a **NN** panic ..

**NNP** TSA **NNP** spokeswoman **NNP** Lisa **NNP** Farbstein **VBD** said **DT** the **NNS** dogs **VBZ** undergo **CD** 12 **NN** weeks **IN** of **NN** training **, ,** **WDT** which **VBZ** costs **IN** about **NN** $200,000 **, ,** **NN** factoring **IN** in **NN** food **, ,** **NNS** vehicles **CC** and **NNS** salaries **IN** for **NNS** trainers ..

**NNS** weeks **IN** of **NN** training **, ,** **WDT** which **VBZ** costs **IN** about **NN** $200,000 **, ,** **NN** factoring **IN** in **NN** food **, ,** **NNS** vehicles **CC** and **NNS** salaries **IN** for **NNS** trainers ..

**NNS** Doctors **VBG** have **VBN** been **VBN** used **IN** in **NN** cargo **NNS** areas **IN** for **DT** some **NN** time **, ,** **CC** but **VBG** have **RB** just **VBN** been **VBN** introduced **RB** recently **IN** in **NN** passenger **NNS** areas **IN** at **NNP** Newark **CC** and **NNP** JFK **NNS** airports .. **NNP** JFK **VBZ** has **CD** one **NN** dog **CC** and **NNP** Newark **VBZ** has **DT** a **NN** handful **, ,** **NNP** Farbstein **VBD** said ..

**Note:** in the demo, notation used is the standard *POS/word* but when we copy-n-paste, we get *POS word*
### Homework 5

Sample part of Speech Tags (*covered in detail later in the course*)

<table>
<thead>
<tr>
<th>Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>#</td>
<td>Pound sign</td>
</tr>
<tr>
<td>$</td>
<td>Dollar sign</td>
</tr>
<tr>
<td>&quot;</td>
<td>Open double quote</td>
</tr>
<tr>
<td>'</td>
<td>Close single quote</td>
</tr>
<tr>
<td>,</td>
<td>Comma</td>
</tr>
<tr>
<td>:</td>
<td>Colon, semi-colon</td>
</tr>
<tr>
<td>-LRB-</td>
<td>Left bracket</td>
</tr>
<tr>
<td>-RRB-</td>
<td>Right bracket</td>
</tr>
<tr>
<td>CC</td>
<td>Coordinating conjunction</td>
</tr>
<tr>
<td>CD</td>
<td>Cardinal number</td>
</tr>
<tr>
<td>DT</td>
<td>Determiner</td>
</tr>
<tr>
<td>EX</td>
<td>Existential there</td>
</tr>
<tr>
<td>FW</td>
<td>Foreign word</td>
</tr>
<tr>
<td>JJ</td>
<td>Adjective</td>
</tr>
<tr>
<td>JJR</td>
<td>Comparative adjective</td>
</tr>
<tr>
<td>JJS</td>
<td>Superlative adjective</td>
</tr>
<tr>
<td>LS</td>
<td>List Item Marker</td>
</tr>
<tr>
<td>MD</td>
<td>Modal</td>
</tr>
<tr>
<td>NN</td>
<td>Singular noun</td>
</tr>
<tr>
<td>NNS</td>
<td>Plural noun</td>
</tr>
<tr>
<td>NNP</td>
<td>Proper singular noun</td>
</tr>
<tr>
<td>NNPS</td>
<td>Proper plural noun</td>
</tr>
<tr>
<td>PDT</td>
<td>Predeterminer</td>
</tr>
<tr>
<td>POS</td>
<td>Possessive ending</td>
</tr>
<tr>
<td>PRP</td>
<td>Personal pronoun</td>
</tr>
<tr>
<td>PP$</td>
<td>Possessive pronoun</td>
</tr>
<tr>
<td>RB</td>
<td>Adverb</td>
</tr>
<tr>
<td>RBR</td>
<td>Comparative adverb</td>
</tr>
<tr>
<td>RBS</td>
<td>Superlative Adverb</td>
</tr>
<tr>
<td>RP</td>
<td>Particle</td>
</tr>
<tr>
<td>SYM</td>
<td>Symbol</td>
</tr>
<tr>
<td>TO</td>
<td>to</td>
</tr>
<tr>
<td>UH</td>
<td>Interjection</td>
</tr>
<tr>
<td>VB</td>
<td>Verb, base form</td>
</tr>
<tr>
<td>VBD</td>
<td>Verb, past tense</td>
</tr>
<tr>
<td>VBG</td>
<td>Verb, gerund/present participle</td>
</tr>
<tr>
<td>VBN</td>
<td>Verb, past participle</td>
</tr>
<tr>
<td>VBP</td>
<td>Verb, non 3rd ps. sing. present</td>
</tr>
<tr>
<td>VBZ</td>
<td>Verb, 3rd ps. sing. present</td>
</tr>
<tr>
<td>WDT</td>
<td>wh-determiner</td>
</tr>
<tr>
<td>WP</td>
<td>wh-pronoun</td>
</tr>
<tr>
<td>WPS</td>
<td>Possesive wh-pronoun</td>
</tr>
</tbody>
</table>
Homework 5

- To make things easier for debugging, I've supplied sample.txt (with the '/'s put back in)
Homework 5

• Extract the POS tags:

```perl
1 open($fh, $ARGV[0]) or die "$ARGV[0] not found!\n";
2
3 while ($line = <$fh>) {
4   $pos_line = ""
5   while ($line =~ m!\S+/\S+!g) {
6      $pos_line .= "$1"
7   }
8   $pos_line = substr $pos_line,1;
9   print "$pos_line\n"
10 }
```

Note: the POS tagger may mistag some of the words

Here: "patrol"

• Result:

- NNPS MD NN DT JJ JJ NN IN NNP NNP POS NNP NNP NNP, IN NNP VBN IN NNP NNP JJ TO VB VBN IN DT JJ NN VBZ VB DT VBN NN.
- IN IN, JJ NNS MD NN DT NNS CC NNS WDT VBP VBN TO VB RB CD IN DT JJ NNS TO NNP/Sunday/POS NNP NNP IN DT NNP NNS CC NNP NNP.
- DT NNP NNP NNP VBD PRP VBZ VBN IN CD NN NNS TO VB NNS VBG RP CC RP IN DT NN IN DT NNP NNP.
- IN NNP, NNP NNS VBD WRB DT NNS MD VB RP JJ JJ NNS IN NNS. RB PRP VBP, PRP VBP VBN TO VB RB IN NN, RB RB RB TO VB NN CC VB DT NN.
- NNP NN NNP NNP VBD DT NNS VBP CD NNS IN NN, WDT VBZ IN NN, NN IN NN, NNS CC NNS IN NNS.
- NNS VBP VBN VBN IN NN NNS IN DT NN, CC VBP RB VBN VBN RB IN NN NNS IN NNP CC NNP NNS. NNP VBZ CD NN CC NNP VBZ DT NN, NNP VBD.
Homework 5

- Be careful: develop the regex slowly
  - e.g. DT/\S+ (JJ/\S+)* NNS?/\S+ (spacing problem)
  - vs DT/\S+ (JJ/\S+) * NNS?/\S+
  - DT = determiner
  - JJ = adjective
  - NN = common noun
  - NNS = plural common noun

DT/the JJ/temporary JJ/no-fly NN/zone
DT/an JJ/unauthorized NN/aircraft
Homework 5

• Submit:
  – 1. Your program
  – 2. Explain your (hard-to-read) regular expression
  – 3. Output in (something like) the following format
    • one NP per line
    • Plus the original line (see next slide for an example)
  – 4. Explain what problems you might encounter
Homework 5 Sample Output

• Example:
  1. DT/The NNP/Transportation NNP/Security NNP/Administration
  2. CD/two NN/dozen NNS/dogs
  3. NNS/passengers
  4. DT/the NN/airport
  5. DT/the NNP/Super NNP/Bowl

– Line: DT/The NNP/Transportation NNP/Security NNP/Administration VBD/said PRP/it VBZ/has VBN/added IN/about CD/two NN/dozen NNS/dogs TO/to VB/monitor NNS/passengers VBG/coming RP/in CC/and RP/out IN/of DT/the NN/airport IN/around DT/the NNP/Super NNP/Bowl ./.
Homework 5

• Question 2:
  – 538: must submit
  – 438: extra credit
Homework 5

• Complex NPs can be formed out of "basic"-NPs.
• Example 1:
  – Helicopters will patrol the temporary no-fly zone around New Jersey's MetLife Stadium
  – *around* is a preposition introducing another NP that modifies the first NP
• Example 2:
  – the Denver Broncos and Seattle Seahawks
  – *and* connects two NPs
• Example 3:
  – Sunday's Super Bowl between the Denver Broncos and Seattle Seahawks
  – NP *between* [ NP *and* NP ]
  – recursive!
Homework 5

- Modify your code to capture and print out complex NPs of this sort

Submit:
- 1. Your program
- 2. Explain your *(hard-to-read)* regular expression
- 3. Output (same format as before)
- 4. What kinds of problems will you encounter?

**Hint:**
- to simplify things, you can use
  - `$bnp = qr/..regexp../;`
- and re-use the bnp (possibly compiled) regex in a larger regex, e.g.:
  - `$line =~ m!$bnp IN/S+ $bnp!`
Zipf's Law

- Zipf's law states that given some corpus of natural language utterances, the frequency of any word is inversely proportional to its rank in the frequency table.
- See:
  - Brown Corpus (1,015,945 words): only 135 words are needed to account for half the corpus.

<table>
<thead>
<tr>
<th>Word</th>
<th>Instances</th>
<th>% Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>The</td>
<td>69970</td>
<td>6.8872</td>
</tr>
<tr>
<td>of</td>
<td>36410</td>
<td>3.5839</td>
</tr>
<tr>
<td>and</td>
<td>28854</td>
<td>2.8401</td>
</tr>
<tr>
<td>to</td>
<td>26154</td>
<td>2.5744</td>
</tr>
<tr>
<td>a</td>
<td>23363</td>
<td>2.2996</td>
</tr>
<tr>
<td>in</td>
<td>21345</td>
<td>2.1010</td>
</tr>
<tr>
<td>that</td>
<td>10594</td>
<td>1.0428</td>
</tr>
</tbody>
</table>

http://www.learnholistically.it/esp-clil/wfk2.htm

[Log – Log scale: straight line](https://finnaarupnielsen.wordpress.com/2013/10/22/zipf-plot-for-word-counts-in-brown-corpus/)
Character Frequency Counting

- Sample code is rather interesting (-e flag):

```
# This example counts character frequencies in a line:

1. $x = "Bill the cat";
2. $x =~ s/(.)/$chars{$_}++;$1/eg;  # final $1 replaces char with itself
3. print "frequency of \$_ is \$chars{\$_}\n"
   foreach (sort {\$chars{\$b} <=> \$chars{\$a}} keys %chars);
4.
```

- Slightly modified but easier to read:

```
1 $x= "This is a slightly simplified version of a rather complicated piece of Perl code."
2 $x =~ s/(.)/\$chars{lc($1)}++;$1/eg;
3 @sorted = sort {\$chars{\$b} <=> \$chars{\$a}} keys %chars;
4 print $x, \"\n\"
5 foreach $key (@sorted) {
6       print "freq of $key is $chars{$key}\n"
7   }
```
Character Frequency Counting

• Output:

```
dhcp-10-142-131-135:ling538-15 sandiway$ perl charfreq.perl
This is a slightly simplified version of a rather complicated piece of Perl code.
  freq of  is 13
  freq of i is 9
  freq of e is 8
  freq of s is 5
  freq of l is 5
  freq of o is 5
  freq of a is 4
  freq of t is 4
  freq of r is 4
  freq of c is 4
  freq of p is 4
  freq of h is 3
  freq of d is 3
  freq of f is 3
  freq of m is 2
  freq of y is 1
  freq of g is 1
  freq of v is 1
  freq of n is 1
  freq of . is 1
```

Does Zipf's Law apply to character frequencies?
Prime Number Testing using Perl Regular Expressions

• Another example:
  – the set of prime numbers is not a regular language
  – $L_{\text{prime}} = \{2, 3, 5, 7, 11, 13, 17, 19, 23, .. \}$

[Prime number - Wikipedia, the free encyclopedia](en.wikipedia.org/wiki/Prime_number)
A prime number (or a prime) is a natural number greater than 1 that has no positive divisors other than 1 and itself. A natural number greater than 1 that is not a ...

Turns out, we can use a Perl regex to determine membership in this set .. and to factorize numbers

/^(11+?)\1+$/
Prime Number Testing using Perl

Regular Expressions

- $L = \{1^n \mid n \text{ is prime}\}$ is not a regular language

- Keys to making this work:
  - \1 backreference
  - Unary notation for representing numbers, e.g.
    - 11111 “five ones” = 5
    - 111111 “six ones” = 6
  - Unary notation allows us to factorize numbers by repetitive pattern matching
    - (11)(11)(11) “six ones” = 6
    - (111)(111) “six ones” = 6
  - Numbers that can be factorized in this way aren’t prime
    - No way to get nontrivial subcopies of 11111 “five ones” = 5
  - Then `/^((11+?)\1+)$/` will match anything that’s greater than 1 that’s not prime
Prime Number Testing using Perl Regular Expressions

- Let’s analyze this Perl regex /^(11+?)\1+$/
- ^ and $ anchor both ends of the strings, forces (11+?)\1+ to cover the string exactly
- (11+?) is non-greedy match version of (11+)
- \1+ provides one or more copies of what we matched in (11+?)

```perl
$n = $ARGV[0];
$string = "1" x $n;

if ($string =~ /^(11+?)\1+$/) {
    print "$n factored using ", length($1), ", not a prime\n"
} else {
    print "$n is a prime\n"
}
```

Question: is the non-greedy operator necessary?
Prime Number Testing using Perl Regular Expressions

Compare `/^(11+?)\1+$/` with `/^(11+)\1+$/`
- i.e. non-greedy vs. greedy matching
- finds smallest factor vs. largest
  - 90021 factored using 3, not a prime (0 secs) vs.
  - 90021 factored using 30007, not a prime (0 secs)
- affects **computational efficiency** for non-primes

**Puzzling behavior:** same output non-greedy vs. greedy
900021 factored using 300007, not a prime (48 secs vs. 13 secs)
Prime Number Testing using Perl Regular Expressions

is of formal (i.e. geeky) interest only...
Prime Number Testing using Perl Regular Expressions

Testing with prime numbers only can take a lot of time to compute...
Prime Number Testing using Perl
Regular Expressions

• `/^\(11+\)\1+$/` vs. `/^(11+)\1+$/`
• i.e. non-greedy vs. greedy matching
• finds smallest factor vs. largest
  – 90021 factored using 3, not a prime (0 secs) vs.
  – 90021 factored using 30007, not a prime (0 secs)

**Puzzling behavior:** same output non-greedy vs. greedy
900021 factored using 300007, not a prime (48 secs vs. 13 secs)
Prime Number Testing using Perl Regular Expressions

- http://www.xav.com/perl/lib/Pod/perlre.html

The following standard quantifiers are recognized:

* Match 0 or more times
+ Match 1 or more times
? Match 1 or 0 times
{n} Match exactly n times
{n,} Match at least n times
{n,m} Match at least n but not more than m times

(If a curly bracket occurs in any other context, it is treated as a regular character.) The ```*``` modifier is equivalent to `{0,}`, the ```+``` modifier to `{1,}`, and the ```?``` modifier to `{0,1}. n and m are limited to integral values less than a preset limit defined when perl is built. This is usually 32766 on the most common platforms.

nearest primes to preset limit

32749  32771
3*32749  32766  3*32771
= 98247  = 98313
Prime Number Testing using Perl Regular Expressions

```perl
$n = $ARGV[0];
$string = "1" x $n;
$t = time();
if ($string =~ /^\d+\d*$/i) {
    $t = time() - $t;
    print "Time $t: $n factored using ", length($1), ", not a prime\n"
} else {
    $t = time() - $t;
    print "Time $t: $n is a prime\n"
}
```

*When preset limit is exceeded: Perl’s regex matching fails quietly*

```
bash-3.2$ perl prime.perl 98247
Time 0: 98247 factored using 3, not a prime
bash-3.2$ perl prime.perl 98313
Time 1: 98313 factored using 32771, not a prime
```
Prime Number Testing using Perl
Regular Expressions

• Can also get non-greedy to skip several factors
• Example: pick non-prime $164055 = 3 \times 5 \times 10937$
  (prime factorization)

```
bash-3.2$ perl prime.perl 164055
Time 0: 164055 factored using $15$, not a prime
bash-3.2$ perl primeg.perl 164055
Time 1: 164055 factored using $54685$, not a prime
```

Because
$3 \times 54685 = 164055$
$5 \times 32811 = 164055$
$32766 \text{ limit}$
$15 \times 10937 = 164055$

Non-greedy: missed factors $3$ and $5$ ...
Prime Number Testing using Perl Regular Expressions

- Results are still right so far though:
  - *wrt.* prime vs. non-prime
- But we predict it will report an incorrect result for
  - 1,070,009,521
  - it should claim (incorrectly) that this is prime
  - since $1070009521 = 32711^2$