Last Time

- Illinois NER System
  - [http://cogcomp.org/page/demo_view/ner]
### Figure 22.1
A list of generic named entity types with the kinds of entities they refer to.

<table>
<thead>
<tr>
<th>Type</th>
<th>Tag</th>
<th>Sample Categories</th>
</tr>
</thead>
<tbody>
<tr>
<td>People</td>
<td>PER</td>
<td>Individuals, fictional characters, small groups</td>
</tr>
<tr>
<td>Organization</td>
<td>ORG</td>
<td>Companies, agencies, political parties, religious groups, sports teams</td>
</tr>
<tr>
<td>Location</td>
<td>LOC</td>
<td>Physical extents, mountains, lakes, seas</td>
</tr>
<tr>
<td>Geo-Political Entity</td>
<td>GPE</td>
<td>Countries, states, provinces, counties</td>
</tr>
<tr>
<td>Facility</td>
<td>FAC</td>
<td>Bridges, buildings, airports</td>
</tr>
<tr>
<td>Vehicles</td>
<td>VEH</td>
<td>Planes, trains, and automobiles</td>
</tr>
</tbody>
</table>

### Figure 22.2
Named entity types with examples.

<table>
<thead>
<tr>
<th>Type</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>People</td>
<td><em>Turing</em> is often considered to be the father of modern computer science.</td>
</tr>
<tr>
<td>Organization</td>
<td>The <em>IPCC</em> said it is likely that future tropical cyclones will become more intense.</td>
</tr>
<tr>
<td>Location</td>
<td>The <em>Mt. Sanitas</em> loop hike begins at the base of <em>Sunshine Canyon</em>.</td>
</tr>
<tr>
<td>Geo-Political Entity</td>
<td><em>Palo Alto</em> is looking at raising the fees for parking in the University Avenue district.</td>
</tr>
<tr>
<td>Facility</td>
<td>Drivers were advised to consider either the <em>Tappan Zee Bridge</em> or the <em>Lincoln Tunnel</em>.</td>
</tr>
<tr>
<td>Vehicles</td>
<td>The updated <em>Mini Cooper</em> retains its charm and agility.</td>
</tr>
</tbody>
</table>
Illinois NER system

• On the ambiguous examples, so-so performance:

The [PER Washington] had proved to be a leaky ship, every passage I made. [PER Washington] was born into slavery on the farm of [PER James Burroughs].

[Loc Washington] went up 2 games to 1 in the 4-game series.

[PER Blair] arrived in [Loc Washington] for what may well be his last state visit.

In June, [PER Washington] passed a primary seat belt law.
Homework 5

- Use the extended NER engine here:
  - [http://cogcomp.org/page/demo_view/NERExtended](http://cogcomp.org/page/demo_view/NERExtended)

<table>
<thead>
<tr>
<th>Key</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>PERSON</td>
<td>Person</td>
<td>ORG</td>
</tr>
<tr>
<td>LOC</td>
<td>Location</td>
<td>TIME</td>
</tr>
<tr>
<td>LAW</td>
<td>Law</td>
<td>NORP</td>
</tr>
<tr>
<td>GPE</td>
<td>Geo-political Entity</td>
<td>LANGUAGE</td>
</tr>
<tr>
<td>PERCENT</td>
<td>Percentage</td>
<td>FAC</td>
</tr>
<tr>
<td>PRODUCT</td>
<td>Product</td>
<td>ORDINAL</td>
</tr>
<tr>
<td>CARDINAL</td>
<td>Cardinal Number</td>
<td>WORK_OF_ART</td>
</tr>
<tr>
<td>MONEY</td>
<td>Money</td>
<td>DATE</td>
</tr>
<tr>
<td>EVENT</td>
<td>Event</td>
<td>QUANTITY</td>
</tr>
</tbody>
</table>
Russian curler Alexander Krushelnitsky has been stripped of his Winter Olympics bronze medal after being found guilty of doping.

The 25-year-old was one of 168 Russians allowed to compete as neutrals at the Games despite the country being banned for previous doping offences.

Krushelnitsky, who won mixed-doubles bronze with wife Anastasia Bryzgalova, tested positive for meldonium.

The Court of Arbitration for Sport said he admitted the anti-doping violation.

With Krushelnitsky and Bryzgalova’s third place annulled, when the bronze medal is officially reallocated it will go to Norway.

The International Olympic Committee banned Russia from this month’s Games in Pyeongchang, South Korea over “systemic” doping at Sochi 2014, which Russia hosted.

But athletes who could prove they were clean were allowed to compete as Olympic athletes from Russia (OAR). The team of 168 was the third biggest behind Canada and the United States.

On Tuesday, the OAR team said a criminal investigation had been opened into Krushelnitsky’s positive test.

It said there was “no evidence of the systematic usage of meldonium”, and that he returned a clear sample on 22 January, when the team’s curlers were tested before flying out to South Korea.

Russian curler Alexander Krushelnitsky has been stripped of his Winter Olympics bronze medal after being found guilty of doping.

The 25-year-old was one of 168 Russians allowed to compete as neutrals at the Games despite the country being banned for previous doping offences.

Krushelnitsky, who won mixed-doubles bronze with wife Anastasia Bryzgalova, tested positive for meldonium.

The Court of Arbitration for Sport said he admitted the anti-doping violation.

With Krushelnitsky and Bryzgalova’s third place annulled, when the bronze medal is officially reallocated it will go to Norway.

The International Olympic Committee banned Russia from this month’s Games in Pyeongchang, South Korea over “systemic” doping at Sochi 2014, which Russia hosted.

But athletes who could prove they were clean were allowed to compete as Olympic athletes from Russia (OAR). The team of 168 was the third biggest behind Canada and the United States.

On Tuesday, the OAR team said a criminal investigation had been opened into Krushelnitsky’s positive test.

It said there was “no evidence of the systematic usage of meldonium”, and that he returned a clear sample on 22 January, when the team’s curlers were tested before flying out to South Korea.
Homework 5

• Evaluate the extended NER's performance on the test sentences.
• In particular:
  1. List the ones it classified wrong
  2. List the named entities it failed to classify at all
  3. For each of 1 and 2, explain how you could classify them correctly, i.e. cite what grammatical evidence you could use from the surrounding text or what piece of world knowledge you needed

• Usual rules of submission:
  • One PDF file only
  • To the TA Patricia Lee by next Wednesday (midnight)
Python's re module

- Regular Expressions in Python: pattern matching
- re module is written in C (programming language)
- https://docs.python.org/3/howto/regex.html

```python
>>> import re
>>> p = re.compile('ab*')
>>> p
re.compile('ab*')
```

<table>
<thead>
<tr>
<th>Regular String</th>
<th>Raw string</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;ab*&quot;</td>
<td>r&quot;ab*&quot;</td>
</tr>
<tr>
<td>&quot;\\section&quot;</td>
<td>r&quot;\\section&quot;</td>
</tr>
<tr>
<td>&quot;\w+\s+\l&quot;</td>
<td>r&quot;\w+\s+\l&quot;</td>
</tr>
</tbody>
</table>
Python's re module

The sequence

```python
prog = re.compile(pattern)
result = prog.match(string)
```

is equivalent to

```python
result = re.match(pattern, string)
```

```python
>>> text = "He was carefully disguised but captured quickly by police."
>>> re.findall(r"\w+ly", text)
['carefully', 'quickly']
```
## Python's re module

<table>
<thead>
<tr>
<th>Method/Attribute</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>match()</code></td>
<td>Determine if the RE matches at the beginning of the string.</td>
</tr>
<tr>
<td><code>search()</code></td>
<td>Scan through a string, looking for any location where this RE matches.</td>
</tr>
<tr>
<td><code>findall()</code></td>
<td>Find all substrings where the RE matches, and returns them as a list.</td>
</tr>
<tr>
<td><code>finditer()</code></td>
<td>Find all substrings where the RE matches, and returns them as an iterator.</td>
</tr>
</tbody>
</table>

```python
>>> import re
>>> p = re.compile('[a-z]+')
>>> m = p.match('tempo')
<_sre.SRE_Match object; span=(0, 5), match='tempo'>
```
Python's re module

<table>
<thead>
<tr>
<th>Method/Attribute</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>group()</td>
<td>Return the string matched by the RE</td>
</tr>
<tr>
<td>start()</td>
<td>Return the starting position of the match</td>
</tr>
<tr>
<td>end()</td>
<td>Return the ending position of the match</td>
</tr>
<tr>
<td>span()</td>
<td>Return a tuple containing the (start, end) positions of the match</td>
</tr>
</tbody>
</table>

```python
>>> m = p.match('tempo')
>>> m
<_sre.SRE_Match object; span=(0, 5), match='tempo'>

>>> m.group()
'tempo'
>>> m.start(), m.end()
(0, 5)
>>> m.span()
(0, 5)
```
Python's re module

```python
>>> p = re.compile(r'\d+')
>>> p.findall('12 drummers drumming, 11 pipers piping, 10 lords a-leaping')
['12', '11', '10']

>>> iterator = p.finditer('12 drummers drumming, 11 ... 10 ... ')
>>> iterator
<callable_iterator object at 0x...>
>>> for match in iterator:
...    print(match.span())
...
(0, 2)
(22, 24)
(29, 31)
```
Python's re module

```python
>>> text = "He was carefully disguised but captured quickly by police."
>>> for m in re.finditer(r"\w+ly", text):
...     print('%02d-%02d: %s' % (m.start(), m.end(), m.group(0)))
07-16: carefully
40-47: quickly
```
Examples from class

```python
$ python3
Python 3.5.2 (v3.5.2:4def2a2901a5, Jun 26 2016, 10:47:25)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import re
>>> text = "Quickly and slowly we go"
>>> re.findall(r"\\w+ly", text)
['Quickly', 'slowly']
>>> text = "Quickly and slowlyied we go"
>>> re.findall(r"\\w+ly", text)
['Quickly', 'slowly']
>>> re.findall(r"\\w+ly\b", text)
['Quickly']
>>> re.findall(r"\\w+ly \", text)
['Quickly ']
>>> text = "Quickly and slowlyied we go simply"
>>> re.findall(r"\\w+ly \", text)
['Quickly ']
>>> re.findall(r"\\w+ly\b", text)
['Quickly', 'simply']
```
Examples from class

```python
>>> text = "Mr. William R. Laidig and Mr. Elton John"
>>> re.findall(r'Mr\.\s+[A-Z][a-z]+\b', text)
['Mr. William', 'Mr. Elton']
>>> re.findall(r'Mr\s+[A-Z][a-z]+\b\s+[A-Z][a-z]+\b', text)
['Mr. Elton John']
>>> re.findall(r'Mr\s+[A-Z][a-z]+\b\s+[A-Z]\s+[A-Z][a-z]+\b', text)
['Mr. William R. Laidig']
>>> text2 = "boy and boys"
>>> re.findall('boys?', text2)
['boy', 'boys']
>>> re.findall(r'Mr\s+[A-Z][a-z]+\b\s+([A-Z]\s+)?[A-Z][a-z]+\b', text)
['R.', '']
>>> re.findall(r'(Mr\s+[A-Z][a-z]+\b\s+([A-Z]\s+)?[A-Z][a-z]+\b)', text)
["'Mr. William R. Laidig', 'R.'], ("Mr. Elton John', '')]
>>> for m in re.finditer(r'(Mr\s+[A-Z][a-z]+\b\s+([A-Z]\s+)?[A-Z][a-z]+\b)', text):
...     print(m.group(0))
...
Mr. William R. Laidig
Mr. Elton John
>>> for m in re.finditer(r'(Mr\s+[A-Z][a-z]+\b\s+([A-Z]\s+)?[A-Z][a-z]+\b)', text):
...     print(m.group(1))
...
Mr. William R. Laidig
Mr. Elton John
>>> for m in re.finditer(r'(Mr\s+[A-Z][a-z]+\b\s+([A-Z]\s+)?[A-Z][a-z]+\b)', text):
...     print(m.group(2))
...
R.
None
```
Examples from class

```python
>>> text3 = "$1,000,000.00 at No. 34"
>>> re.findall(r'\$[\d,]+', text3)
['$1,000,000']
>>> text3 = "$1,000,000.05 at No. 34"
>>> re.findall(r'\$[\d,\.]+', text3)
['$1,000,000.05']
```