LING/C SC/PSYC 438/538

Lecture 13

Sandiway Fong

Today's Topic

- Homework 8 Review
- Last Time:
  - two ways of inserting Perl code into regex
    - `s/regex/`*`code`*`/e`
    - `(?{`*`code`*`})`
- More on powerful features in Perl regex:
  - lookahead
  - lookbehind
- Predicate-Argument Structure
  - *preparing you for Thursday's Homework 9*
  - Framenet
  - Stanford CoreNLP

Homework 8 Review

- In the real world, i.e. *with real datasets*, we can't be absolutely sure:
  - we matched everything we want (**Recall ratio**)
  - we don't have spurious matches (**Precision ratio**)
  - we can't even know what the overall Precision/Recall is
    - *but we can get a sample estimate*

Homework 8 Review

- **Question 1a:**
  - in English, names typically begin with an Upper case letter. Other characters may be lower/upper case or include a hyphen/dash (-), e.g. *ABC–CDE*.
  - Write a regex and find all the matching **words** in the article. How many are there?
- Code:
  - `perl -le 'open $f, "pandora.txt"; while (<$f>) {while (/\b[A-Z][A-Za-z-]*\b/g) {print $&}}' | wc -l`
  - `1097`
- Permit single letter names? If not, `\b[A-Z][A-Za-z-]+\b`
- Gets more than named entities: words at the start of sentence: e.g. *The*
- Doesn't get names beginning with lowercase letter, e.g. *al*–XYZ, *de* or *bin*.

Homework 8 Review

https://www.thefashionlaw.com

As Twitter Becomes X, A Dive into Single Letter Trademarks

TFL

Trademark rights in (and registrations for) a single letter for use on certain goods/services? This is not only possible under trademark law in the U.S., but as it turns out, it is pretty common, as well. "Single letters are among the most popular trademarks registered in the United States," Chris Chafin previously wrote for Fast Co. Back in 2017, each letter of the alphabet was the subject of, at a minimum, hundreds of trademark registrations, he noted. There were, for example, over 2,000 registrations for the letter S, making it the most popular. There were 1,102 registrations for V, 1,100 for E, and 1,816 for A. And those numbers have risen since then, there are currently almost 1,000 registrations for the letter X, as companies flock to and secure trademark registrations for single letters, usually in stylized forms.

Homework 8 Review

- **Question 1b:** last lecture we mentioned use of
  - `open qw(:std :utf8);`
  - Find the differences in the words reported when running your code with this declaration.
  - **Hint**: you may want to think about `[A-Za-z-]` vs `[\w-]`
- Code:
  - `perl -le 'use open qw(:std :utf8); open $f, "pandora.txt"; while (<$f>) {while (/\b[A-Z][A-Za-z-]*\b/g) {print $&}}' | wc -l`
  - `1092 (vs. 1097: Alem Erdo O Piau R)`
  - `perl -le 'use open qw(:std :utf8); open $f, "pandora.txt"; while (<$f>) {while (/\b[A-Z][\w-]*\b/g) {print $&}}' | wc -l`
  - `1097`
  - Alemán Erdoğan Oštro Piauí Rönesans

Homework 8 Review

- **Question 1c:**
  - do all name words begin with an Upper case letter? Find two that don't.
  - `al-Zayanis`
  - `Zayed bin Rashid al-Zayani`
  - `Helena de Chair`
  - `then-President`
  - `1MDB`
  - `maybe others?`

Homework 8 Review

- **Question 2:**
  - abbreviations/acronyms often consist of words, #letters ≥2, containing only Upper case letters, possibly with periods separating them,
  - e.g. TV, US, U.S., TASS.
  - Write a regex for this. How many are there?
- Code:
  - `perl -le 'open $f, "pandora.txt"; while (<$f>) {while (/\b[A-Z\.]{2,}\b/g) {print $&}}' | wc -l`
  - `90`
  - Gets uppercase words too:
    - `WANT MORE STORIES THAT ROCK THE WORLD`

Homework 8 Review

- **Question 3:**
  - many named entities are *n*-grams, *n*≥2, a sequence of words:
    - e.g. `Al Mawarid Bank, British Prime Minister Tony Blair`
  - each beginning with an Upper case letter, **optionally** beginning with a title with leading capitalization:
    - e.g. `Mr(s), Ms, Dr, (Prime) Minister, President` or `King/Queen (of).`
    - e.g. `King of Jordan`
  - Write a regex and find all the matching sequences (*#words* ≥2). Print them. How many are there?
- Code:
  - `perl -le 'use open qw(:std :utf8); open $f, "pandora.txt"; while (<$f>) {while (/\b[A-Z][\w-]*((\s+of)?\s+[A-Z][\w-]*)+/g) {print $&}}' | wc -l`
  - 221
  - `Jackal of Zacapa / House of Commons`

Homework 8 Review

The Pandora Papers

King of Jordan

Czech Republic

British Prime Minister Tony Blair

Russian President Vladimir Putin

United States

French Riviera

Czech Republic

Great Plains

United States

King of Jordan

Arab Spring

Pandora Papers

The International Consortium of Investigative Journalists

Pandora Papers

An ICIJ

British Virgin Islands

Paris-based Organization

Economic Cooperation

WANT MORE STORIES THAT ROCK THE WORLD

The Pandora Papers

Sachin Tendulkar

Claudia Schiffer

Fat One

Sachin Tendulkar

Claudia Schiffer Image

Getty Images

Raffaele Amato

United Kingdom

The Pandora Papers

Pandora Papers

British Virgin Islands

Morgan Stanley

A Morgan Stanley

The Pandora Papers

Baker McKenzie

Baker McKenzie

Ihor Kolomoisky

Baker McKenzie

Jho Low

Baker McKenzie

Hong Kong

Baker McKenzie

Baker McKenzie

The Pandora Papers

Panama Papers

The Panama Papers

Mossack Fonseca

The Pandora Papers

The Pandora Papers

The Washington Post

The Guardian

Radio France

Oštro Croatia

Indian Express

The Standard

Le Desk

Diario El Universo

Persian Gulf

South China Sea

The Pandora Papers

British Virgin Islands

King Abdullah II

King Abdullah II

Jordan Pix

Getty Images

Middle East

Annelle Sheline

Middle East

Pandora Papers

Najib Mikati

Hassan Diab

Riad Salameh

Marwan Kheireddine

Al Mawarid Bank

Pandora Papers

Al Mawarid Bank

Wafaa Abou Hamdan

Imran Khan

Panama Papers

The Panama Papers

Nawaz Sharif

The Guardian

Panama Papers

Pandora Papers

Chaudhry Moonis Elahi

Pandora Papers

Kenyan President Uhuru Kenyatta

Czech Prime Minister Andrej Babis

Czech Prime Minister Andrej Babis

Stefan Wermuth

Getty Images

Chateau Bigaud

In February

Tony Blair Institute

Global Change

Labour Party

West Midlands

The Pandora Papers

British Virgin Islands

The London

Cherie Blair

Cherie Blair

Middle East

The Blairs

Cherie Blair

Robert Palmer

Tax Justice UK

The Guardian

In June

Paulo Guedes

The Pandora Papers

Dreadnoughts International Group

British Virgin Islands

Homework 8 Review

Revista Piauí	Cayman Islands	South Dakota	Nicos Chr	Panama Papers
In December	South Dakota	South Dakota	Pandora Papers	Jacob Rees-Mogg
Some Bahamian	South Dakota	South Dakota	Cyprus President Nicos Anastasiades	British Conservative Party
Latin American	Susan Wismer	Corporate Transparency Act	Leonid Lebedev	House of Commons
South Dakota	Adam Hofri-Winogradow	Yehuda Shaffer	The Cypriot	The Pandora Papers
Dominican Republic	Pandora Papers	The U	Alexander Abramov	Mossack Fonseca
Vice President Carlos Morales Troncoso	The Washington Post	Billionaire Erman Ilicak	President Putin	Iqbal Memon
Sioux Falls	Federico Kong Vielman	The Turkish	Theophanis Philippou	New Delhi
South Dakota	Kong Vielman	Rönesans Holding	Another Russian	Pandora Papers
South Dakota	Sioux Falls	Recep Tayyip Erdoğan	Pandora Papers	Juan Andres Donato Bautista
South Dakota	Carlos Manuel Arana Osorio	Ayse Ilicak	Konstantin Ernst	Presidential Commission
The Pandora Papers	Jackal of Zacapa	British Virgin Islands	Russian TV	Good Government
South Dakota	Guatemala City	Pandora Papers	Konstantin Ernst	British Virgin Island
Trident Trust Co	President Jimmy Morales	Covar Trading Ltd	Artyom Geodakyan	Panama Papers
Sioux Falls	Kong Vielman	Covar Trading	Getty Images	The Philippines
South Dakota	Pasion River	Pandora Papers	The Pandora Papers	President Rodrigo Duterte
Salwan Georges	Nacional Agro Industrial SA	The American	Winter Olympics	Alexander Abramov
The Washington Post	Kong Vielman	Robert F	Mae Buenaventura	Countering America
South Dakota	South Dakota	Robert T	Ferdinand Marcos	Adversaries Through Sanctions Act
The U	Latin American	Glenn Godfrey	The Marcos	
New York-based	South Dakota	A U	An ICIJ	
The U	Guillermo Lasso	Neither CILTrust	Mossack Fonseca	

Homework 8 Review

- **Question 4**:
  - using the Perl hash table described in a previous lecture, re-do **Question 3** and collect together mentions of named entities, e.g. `Baker McKenzie` occurs multiple times. Then print names and number of occurrences in tabular form.
- Code:
  - ```
    perl -le 'use open qw(:std :utf8); open $f,
    "pandora.txt"; while (<$f>) {while (/\b[A-Z][\w-
    ]*((\s+of)?\s+[A-Z][\w-]*)+/g) {$ne{$&}++}}; for
    (sort {$ne{$b} <=> $ne{$a}} (keys %ne)) {print "$_,
    $ne{$_}"}'
    ```
    ```

Homework 8 Review

South Dakota, 14

Pandora Papers, 13

The Pandora Papers, 13

Baker McKenzie, 6

British Virgin Islands, 6

Panama Papers, 5

Getty Images, 4

The Washington Post, 3

The Guardian, 3

Cherie Blair, 3

Middle East, 3

The U, 3

Mossack Fonseca, 3

Sioux Falls, 3

Kong Vielman, 3

King Abdullah II, 2

King of Jordan, 2

An ICIJ, 2

Latin American, 2

United States, 2

Czech Republic, 2

Konstantin Ernst, 2

Sachin Tendulkar, 2

Alexander Abramov, 2

Al Mawarid Bank, 2

Czech Prime Minister Andrej Babis, 2

The Panama Papers, 2

Pasion River, 1

Mae Buenaventura, 1

Oštro Croatia, 1

Carlos Manuel Arana Osorio, 1

Adam Hofri-Winogradow, 1

Hassan Diab, 1

Iqbal Memon, 1

Jacob Rees-Mogg, 1

Radio France, 1

The Cypriot, 1

Annelle Sheline, 1

A Morgan Stanley, 1

Ayse Ilicak, 1

Najib Mikati, 1

British Virgin Island, 1

Le Desk, 1

Yehuda Shaffer, 1

Nicos Chr, 1

In February, 1

Ihor Kolomoisky, 1

Arab Spring, 1

The Philippines, 1

Artyom Geodakyan, 1

Adversaries Through Sanctions Act, 1

Regex Lookahead and Lookbehind

- We've already seen some **zero-width** regexs:
  - ^ (start of string)
  - $ (end of string)
  - \b (word boundary)
    - matches the imaginary position between \w\W or \W\w, or just before beginning of string if ^\w, just after the end of the string if \w$
- zero-width because position of match (so far), pos, doesn't change!
  1. (?=*regex*) (lookahead from current position)
  2. (?<=*regex*) (lookbehind from current position)
  3. (?!*regex*) (negative lookahead)
  4. (?<!*regex*) (negative lookbehind)

Lookahead (and lookbehind)

Lookaround Assertions

Lookaround assertions are zero-width patterns which match a specific pattern without including it in `$&`. Positive assertions match when their subpattern matches, negative assertions match when their subpattern fails. Lookbehind matches text up to the current match position, lookahead matches text following the current match position.

`(?=pattern)`

A zero-width positive lookahead assertion. For example, `/\w+(?=\t)/` matches a word followed by a tab, without including the tab in `$&`.

`(?!pattern)`

A zero-width negative lookahead assertion. For example `/foo(?!bar)/` matches any occurrence of "foo" that isn't followed by "bar". Note however that lookahead and lookbehind are NOT the same thing. You cannot use this for lookbehind.

If you are looking for a "bar" that isn't preceded by a "foo", `/(?!foo)bar/` will not do what you want. That's because the `(?!foo)` is just saying that the next thing cannot be "foo"--and it's not, it's a "bar", so "foobar" will match. Use lookbehind instead (see below).

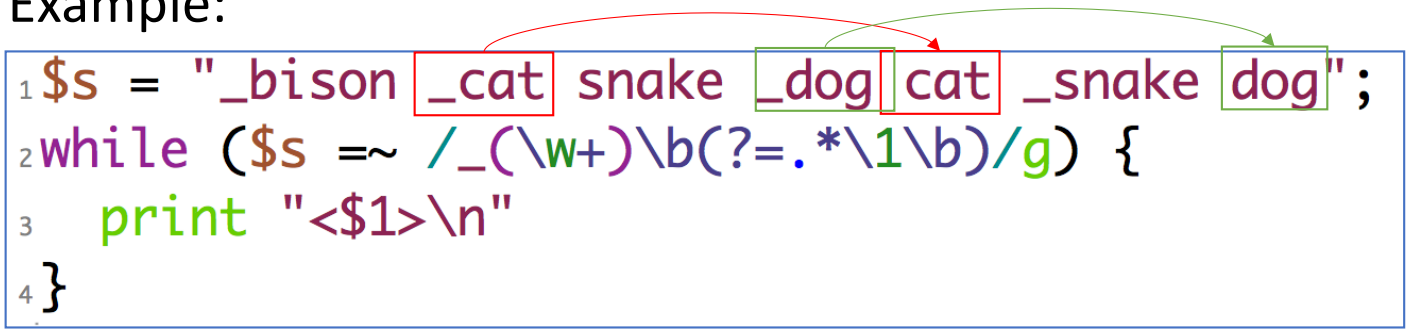
`/(?<=\t)\w+/`

`(?<=pattern)` lookbehind for *pattern*

`(?<!pattern)` negative lookbehind for *pattern*

Regex Lookahead and Lookbehind

- Example:



```
1 $s = "_bison _cat snake _dog cat _snake dog";
2 while ($s =~ /_(\w+)\b(?=.*\1\b)/g) {
3   print "<$1>\n"
4 }
```

```
$perl test.perl
<cat>
<dog>
$
```

looks for a word beginning with _ such that there is a duplicate ahead (without the _)
(?= ..) means **lookahead**

Regex Lookahead and Lookbehind

Some restrictions apply:

lookbehind (*in most versions of Perl*) cannot be of variable length

- From `perlretut`:
  - **Lookahead** can match arbitrary regexps,
  - but **lookbehind** prior to 5.30 `(?<=fixed-regexp)` only works for regexps of fixed width, *i.e.*, a fixed number of characters long. Thus `(?<=(ab|bc))` is fine, but `(?<=(ab)*)` prior to 5.30 is not.

Debugging Perl regex

- **(?{** *Perl code* **})** can be inserted anywhere in a regex
- can assist with debugging
- **Example**:

```
1 $s = "_bison _cat snake _dog cat _snake dog";
2 while ($s =~ /_(\w+)\b(?{print "$1\n"})(?=.*\1\b)/g) {
3   print "<$1>\n"
4 }
```

```
$perl test.perl
bison
cat
<cat>
dog
<dog>
snake
```

Regex Lookahead and Lookback

- `(?<!pattern)` `/(?<!bar)foo/`
- `(*nlb:pattern)`
- `(*negative_lookbehind:pattern)`

  A zero-width negative lookbehind assertion. For example `/(?<!bar)foo/` matches any occurrence of "foo" that does not follow "bar".

  Prior to Perl 5.30, it worked only for fixed-width lookbehind, but starting in that release, it can handle variable lengths from 1 to 255 characters as an experimental feature. The feature is enabled automatically if you use a variable length lookbehind assertion, but will raise a warning at pattern compilation time, unless turned off, in the `experimental::vlb` category. This is to warn you that the exact behavior is subject to change should feedback from actual use in the field indicate to do so; or even complete removal if the problems found are not practically surmountable. You can achieve close to pre-5.30 behavior by fatalizing warnings in this category.

Background

- Background stuff you should familiar yourself with …
  - Predicate-argument structure
  - Stanford CoreNLP

Background

Predicate-Argument Structure (typically for verbs)

- Example
  - *John saw/noticed the javelina*
  - *notice*(experiencer, theme) or *see*(experiencer, theme)
  - John noticed that Mary saw the javelina
  - *notice*(perceiver, proposition) 1st argument: subject, 2nd argument: direct object
  - the cat chased the mouse
  - *chase*(agent, theme) 1st argument: subject, 2nd argument: direct object
  - the mouse was chased by the cat (*passivization*)
  - *John was jogged for an hour (**passivization*)
  - John jogged for an hour (intransitive)
  - *jog*(agent)

Background

- Different representations exist in the literature.
- Simple:
  - *John saw/noticed the javelina*
  - *notice*(experiencer, theme)
- Neo-Davidsonian:
  - event(*e*) & experiencer(*e*, John) & theme(*e*, javelina)

Background

- Framenet
  - https://framenet.icsi.berkeley.edu/fndrupal/luIndex
  - Words in this frame have to do with a Cognizer adding some Phenomenon to their model of the world.

Lexical Entry

notice.v

Frame: Becoming_aware

Definition:

COD: become aware of.

Frame Elements and Their Syntactic Realizations

The Frame Elements for this word sense are (with realizations):

Frame Element	Number Annotated	Realization(s)
Cognizer	(25)	CNI.-- (3) NP.Ext (22)
Ground	(7)	PP[from].Dep (1) PP[in].Dep (2) PP[at].Dep (3) PP[on].Dep (1)
Phenomenon	(24)	NP.Obj (16) Sfin.Dep (7) Sinterrog.Dep (1)
State	(3)	INI.-- (2) VPing.Dep (1)
Time	(2)	PP[on].Dep (1) AVP.Dep (1)

core

core

Background

Framenet Examples:

- 420-that-sfin
  1. [Cognizer I] soon *NOTICED* [Phenomenon that the car was being driven very dangerously] .
  2. Then off they went but [Cognizer I] had *NOTICED* [Phenomenon that Mrs Taylor was really crying] .
  3. [Cognizer You] will *NOTICE* that there is , [Ground in the wording of that letter] , [Phenomenon something curious] .
- 430-sfin
  1. *NOTICE* [Phenomenon the street names] [Ground in the centre of Bristol] .[Cognizer CNI]
  2. [Cognizer You] may *NOTICE* [Phenomenon that food tastes different when you are pregnant] .
  3. ` I do n't suppose [Cognizer anyone] will even *NOTICE* [Phenomenon you 're not there] .
  4. [Cognizer Nobody] even *NOTICED* [Phenomenon I was in the room !]
- 480-swh
  1. On the way [Cognizer he] *NOTICED* [Phenomenon how quiet the school seemed] .
- 520-np-vping
  1. ` Did [Cognizer you] *NOTICE* [Phenomenon any knives] [State lying about] ? "
- 570-np-ppabout
  1. ` I see [Cognizer you] have *NOTICED* [Phenomenon a certain peculiarity about my appearance .] "
- 570-np-ppat
  1. When examining the wound , [Cognizer I] *NOTICED* [Phenomenon a dark area] [Ground at each end of the cut] .
  2. [Cognizer Users of the main car park at Park Royal] will have *NOTICED* [Phenomenon a new fence] [Ground at the back of the site] .
  3. Then [Cognizer I] *NOTICED* [Phenomenon Alec] [Ground at the other end of the bench] .

Background

Lexical Units:

- *chance (across).v, chance (on).v, come (across).v, come (upon).v, descry.v, detect.v, discern.v, discover.v, discovery.n, encounter.v, espy.v, fall (on).v, find (oneself).v, find out.v, find.v, happen (on).v, learn.v, locate.v, note.v, notice.v, observe.v, perceive.v, pick up.v, recognize.v, register.v, spot.v, spy out.v, tell.v*

Not present **Perception_experience** verbs:

- *detect.v, experience.n, experience.v, feel.v, hear.v, overhear.v, perceive.v, perception.n, see.v, sense.v, smell.v, taste.v, witness.v*

Not present **Perception_active** verbs:

- *admire.v, attend.v, eavesdrop.v, eye.v, feel.v, gape.v, gawk.v, gaze.n, gaze.v, glance.n, glance.v, goggle.v, listen.v, look.n, look.v, observation.n, observe.v, palpate.v, peek.n, peek.v, peep.v, peer.v, savour.v, smell.v, sniff.n, sniff.v, spy.v, squint.v, stare.n, stare.v, taste.n, taste.v, view.v, watch.v*

Background

- Unified Verb Index
  - https://verbs.colorado.edu/verb-index/vn3.3/

notice — SEE-30.1-1-1, (PROPBANK), (FN BECOMING_AWARE), (GROUPING)

FRAMES	
NP V S	
EXAMPLE	"I saw her bake the cake."
SYNTAX	EXPERIENCER V STIMULUS <+OC_BARE_INF>
SEMANTICS	PERCEIVE(DURING(E), EXPERIENCER, STIMULUS) IN_REACTION_TO(E, STIMULUS)
NP V S_ING	
EXAMPLE	"I saw him laughing."
SYNTAX	EXPERIENCER V STIMULUS <+OC_ING>
SEMANTICS	PERCEIVE(DURING(E), EXPERIENCER, STIMULUS) IN_REACTION_TO(E, STIMULUS)
NP V S_ING	
EXAMPLE	"I saw their laughing and joking."
SYNTAX	EXPERIENCER V STIMULUS <+POSS_ING>
SEMANTICS	PERCEIVE(DURING(E), EXPERIENCER, STIMULUS) IN_REACTION_TO(E, STIMULUS)

Background: Propbank

Roleset id: notice.01 , *become aware of*, Source: , vncls: , framnet:

notice.01: NOTICE-V NOTES: Frames file for 'notice' based on sentences in wsj. Verb
30.1-1. Framed by Katie. (from notice.01-n) NOTICING-N, TAKE_NOTICE-L NOTE

Aliases:

Alias	Frame
notice (v.)	
notice (n.)	
noticing (n.)	
take_notice (l.)	

Roles:

Arg0-PPT: *noticer* (vnrole: 30.1-1-Experiencer)
Arg1-PAG: *noticed* (vnrole: 30.1-1-Stimulus)

- Propbank:
  - • ARGn-PAG … proto-agent
    - ARGn-PPT … proto-patient

Background: Propbank

notice-v; 2 Senses

- **Sense Number 1: observe, perceive or become aware of something**
- Examples:
  Did you notice what he had in his hand?
  I noticed that he avoided mentioning her name.
  Mary waved at the man but he didn't seem to notice.
  Starting in 1987, scientists noticed large drops in the amount of phytoplankton.
  Her musical talent was first noticed by the critics at the age of 12.
- Mappings:
  VerbNet: see-30.1-1-1
  FrameNet: Becoming_aware
  PropBank: notice.01
  WordNet 3.0 Sense Numbers: 1, 2, 4

Background: Propbank

notice-v; 2 Senses

- **Sense Number 2: bring to attention; give notice or announce**
- Examples:
  The Solicitor General noticed the court of a change in Justice Department police.
  The foundation noticed the Council of the new approach.
- Mappings:
  VerbNet: NM
  FrameNet: NM
  PropBank: NM

Background

Predicate-Argument Structure (typically for verbs)

- Example
  - *the librarian put the book
  - the librarian put the book on the table
  - *put*(agent, theme, location)
  - Mary gave John the textbook
  - *Mary gave John
  - *give*(agent, goal, theme)
  - Mary gave the textbook to John

Background: Framenet

give:

Core:

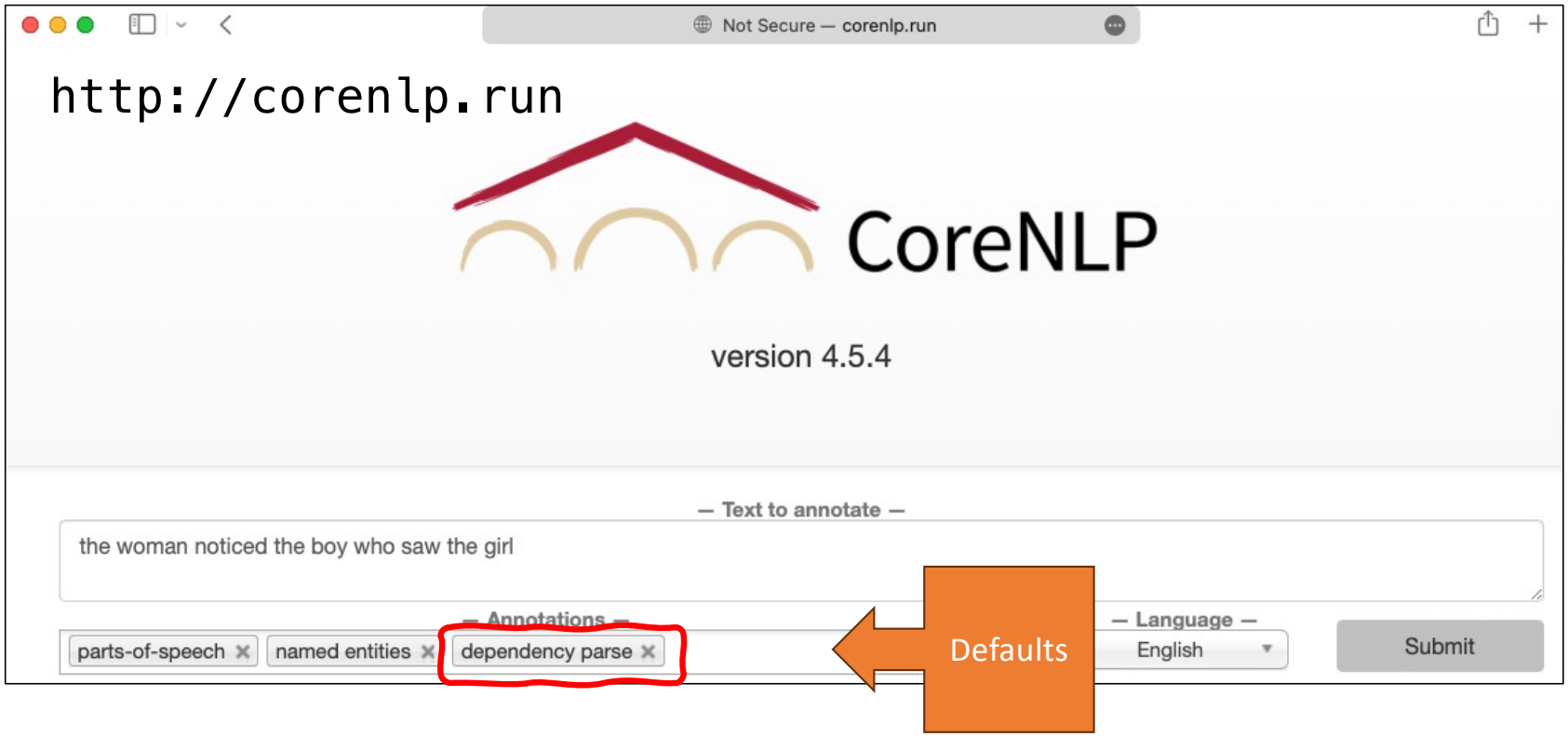
Donor [Donor]	The person that begins in possession of the Theme and causes it to be in the possession of the Recipient .
Recipient [Rec]	The entity that ends up in possession of the Theme.
Theme [Thm] **Semantic Type:** Physical_object	The object that changes ownership.

put:

Core:

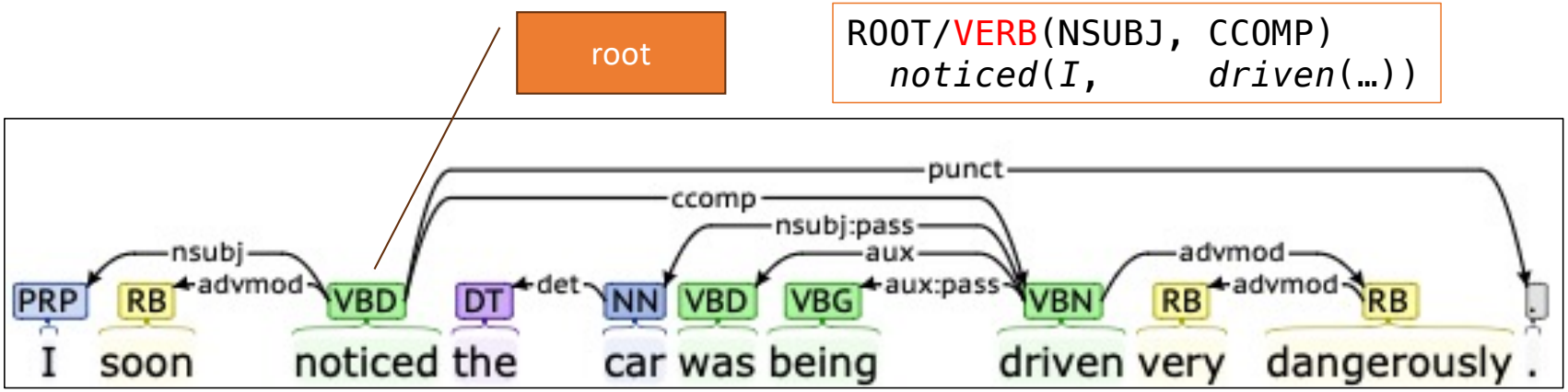
Agent [Agt] **Semantic Type:** Sentient	The Agent is the person (or other force) that causes the Theme to move. The waiter PLACED the food on the table.
Cause [Cause] **Excludes:** Agent	Grass , which is sown with clover , provides rich pasture for cattle in summer and the clover is another plant which PUTS nitrogen into the soil .
Goal [Goal] **Semantic Type:** Goal	The FE Goal is the location where the Theme ends up. This FE is profiled by words in this frame. The waiter PLACED the food on the table.
Theme [Thm] **Semantic Type:** Physical_object	The Theme is the object that changes location during the Placing. The waiter PLACED the food on the table.

Background: CoreNLP



Background: CoreNLP

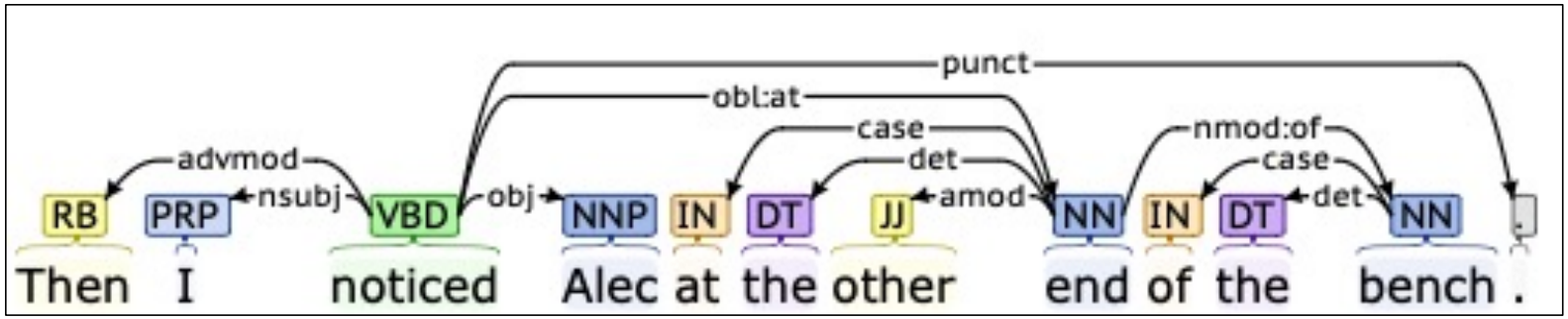
- Examples (from Framenet):
  1. [$_{\text{Cognizer}}$I] soon *NOTICED* [$_{\text{Phenomenon}}$ the car was being driven very dangerously] .
  2. Then [$_{\text{Cognizer}}$I] *NOTICED* [$_{\text{Phenomenon}}$Alec] [$_{\text{Ground}}$at the other end of the bench] .



Background: CoreNLP

- Examples (from Framenet):
  1. [$_{\text{Cognizer}}$I] soon *NOTICED* [$_{\text{Phenomenon}}$ the car was being driven very dangerously] .
  2. Then [$_{\text{Cognizer}}$I] *NOTICED* [$_{\text{Phenomenon}}$Alec] [$_{\text{Ground}}$at the other end of the bench] .

```
ROOT/VERB(NSUBJ, OBJ)
  noticed(I,     Alec)
```



Background: Stanford Dependencies

- Some definitions you may find useful
  https://nlp.stanford.edu/software/dependencies_manual.pdf
  - ***ccomp***: clausal complement
    A clausal complement of a verb or adjective is a dependent clause
  - ***dobj***: direct object
    The direct object of a VP is the noun phrase which is the (accusative) object of the verb.
  - ***iobj***: indirect object
    The indirect object of a VP is the noun phrase which is the (dative) object of the verb.
  - ***nsubj***: nominal subject
    A nominal subject is a noun phrase which is the syntactic subject of a clause.
  - ***rcmod***: relative clause modifier
    A relative clause modifier of an NP is a relative clause modifying the NP. The relation points from the head noun of the NP to the head of the relative clause, normally a verb.
  - ***vmod***: reduced non-finite verbal modifier
    A reduced non-finite verbal modifier is a participial or infinitive form of a verb heading a phrase (which may have some arguments, roughly like a VP).

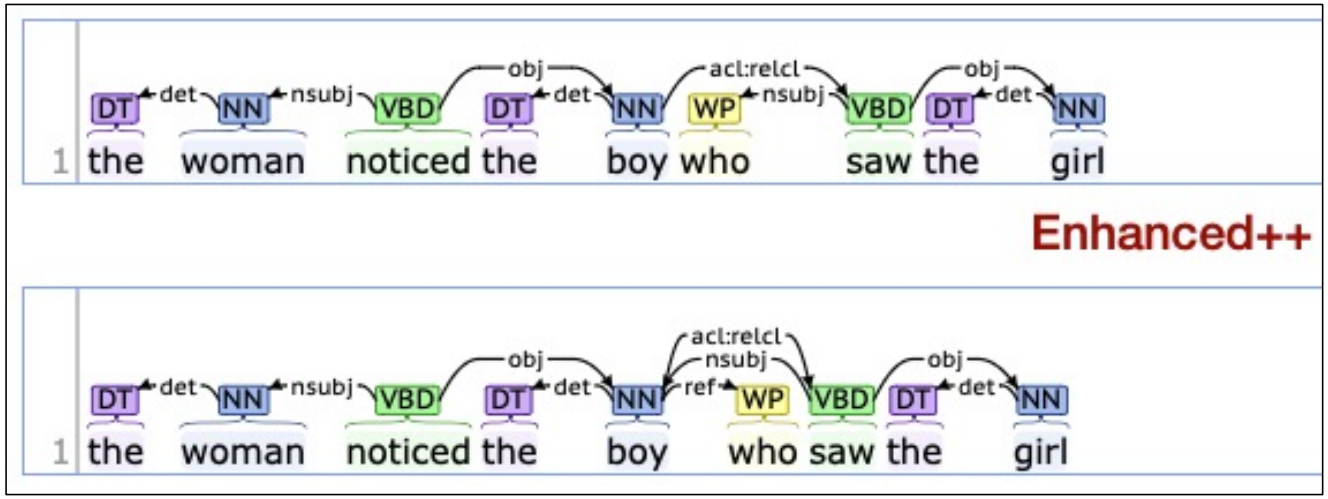
Background: Universal Dependencies

https://universaldependencies.org/u/dep/index.html

	Nominals	Clauses	Modifier words	Function Words
Core arguments	nsubj obj iobj	csubj ccomp xcomp		
Non-core dependents	obl vocative expl dislocated	advcl	advmod* discourse	aux cop mark
Nominal dependents	nmod appos nummod	acl	amod	det clf case
Coordination	**MWE**	**Loose**	**Special**	**Other**
conj cc	fixed flat compound	list parataxis	orphan goeswith reparandum	punct root dep

Background: CoreNLP

- Root: noticed(woman, boy)
- ACL:RELCL points back to NOUN boy
- ACL:RELCL/VERB(NSUBJ/PRON, OBJ)
- We infer saw(boy, girl)



Background: Universal Dependencies

- acl = adnominal clause (basically, a sentence that modifies a noun)

`acl:relcl`: relative clause modifier

A relative clause modifier of a nominal is a clause that modifies the nominal, whereas the nominal is coreferential with a constituent inside the relative clause (here the constituent may be realized as a relative pronoun, another relative word, or it may not be overtly realized at all). The `acl:relcl` relation points from the head of the modified nominal to the head of the relative clause.

Depending on language, it may be required that relative clauses are finite. For example, English non-finite clauses are traditionally not termed relative; therefore, *the girl* ***that was born today*** is a relative clause because it is finite, while *the girl* ***born today*** is non-finite (the participle is not accompanied by a finite auxiliary) and it uses the plain `acl` relation. In other languages however, the distinction between finite and non-finite clauses may not exist or may not be used as a criterion for relative clauses.