LING/C SC/PSYC 438/538

Lecture 14

Sandiway Fong

Today's Topic

- Last Time:
  - inserting Perl code into regex
  - `s/regex/`*`code`*`/e`
  - `(?{`*`code`*`})`
- More on powerful features in Perl regex:
  - lookahead
  - lookbehind
- Predicate-Argument Structure
- Google Natural Language
- Homework 9: Parts 1 and 2:
  - given out early, *not due until Sunday midnight*

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).

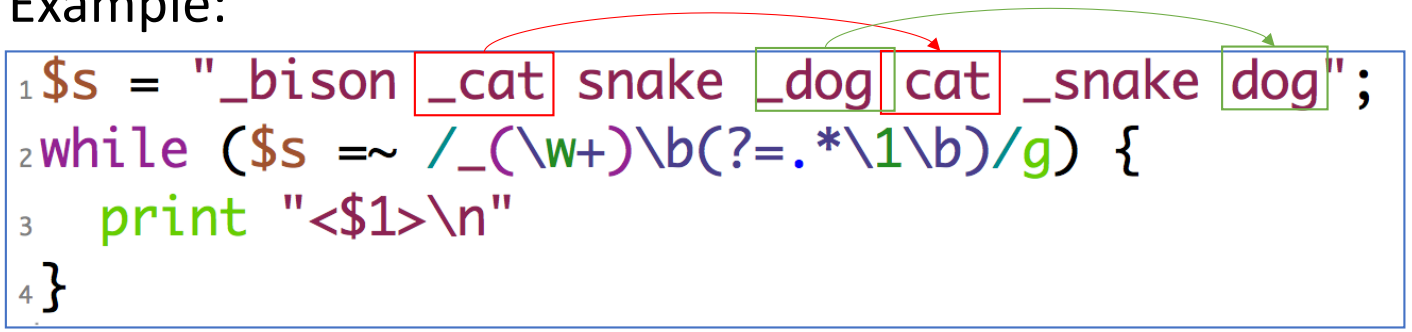
`/(?<=\t)\w+/`

`(?<=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)`
- `(*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.

```
/(?<!bar)foo/
```

Homework 9: Part 1

- Background stuff you should familiar yourself with …
  - predicate-argument structure
  - Google Natural Language

Background: Part 1

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: Part 1

- 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: Part 1

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: Part 1

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: Part 1

- 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: Part 1

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: Part 1

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

- **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: Part 1

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: Part 1

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.

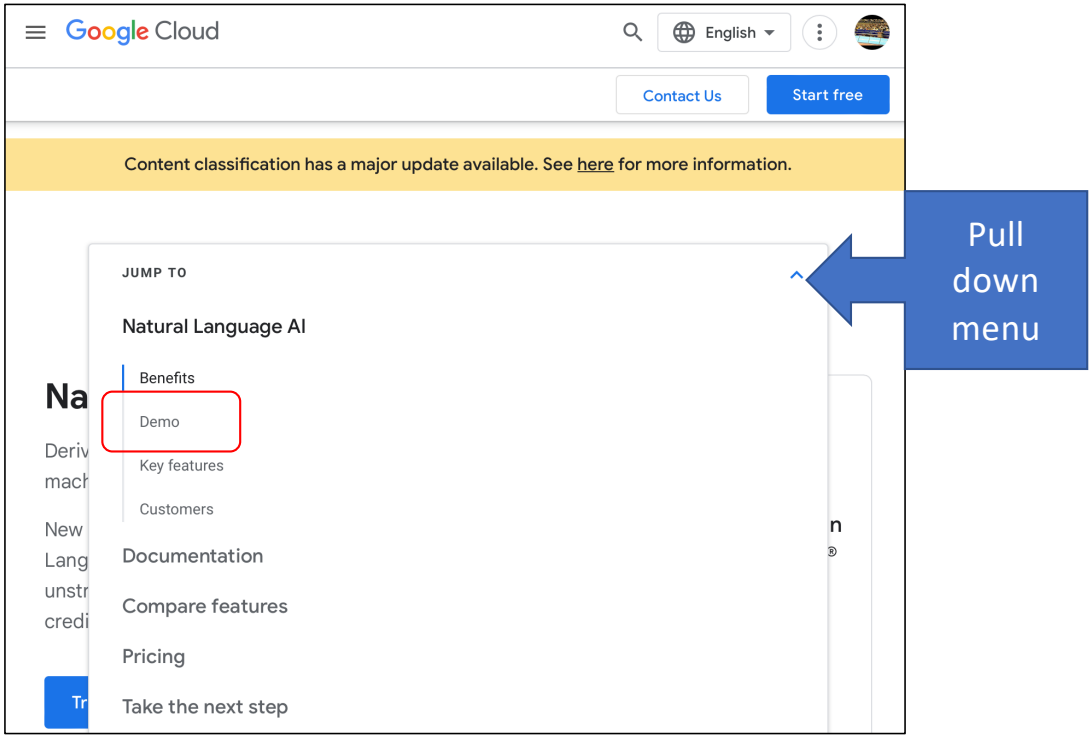
put:

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.

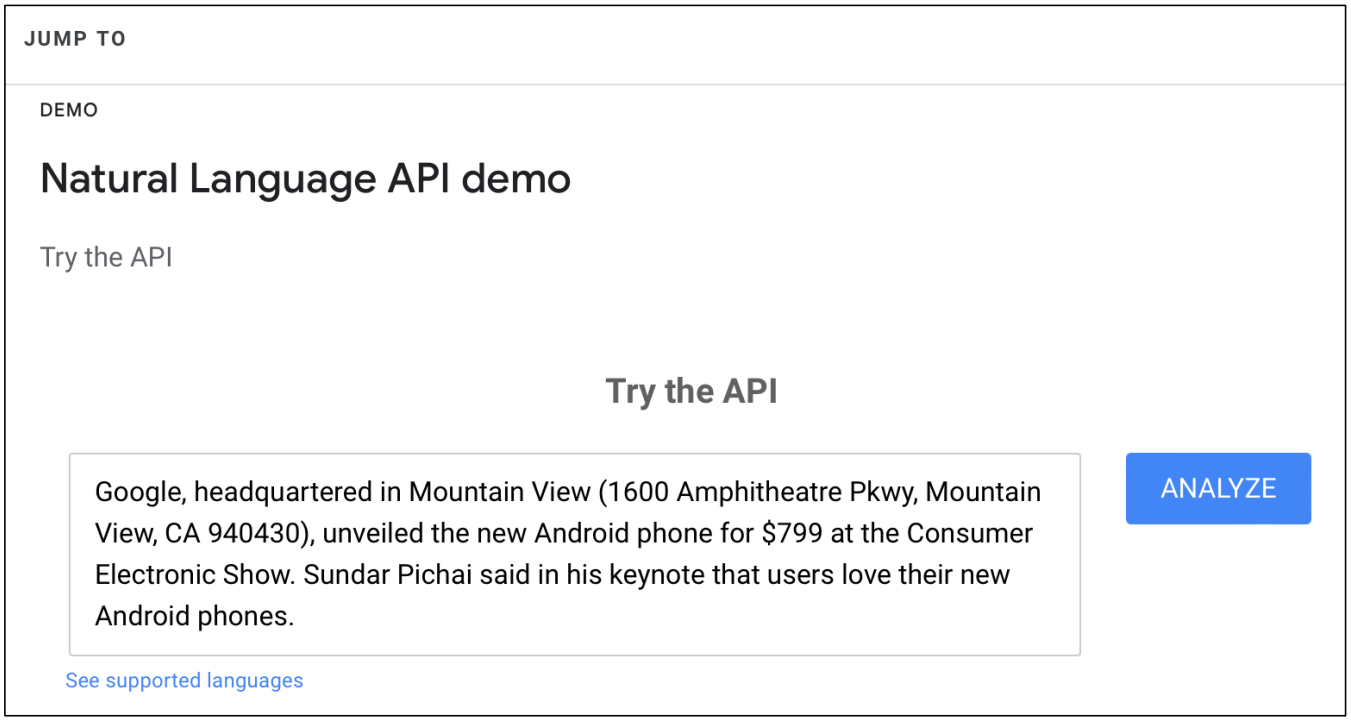
Background: Part 1

https://cloud.google.com/natural-language/



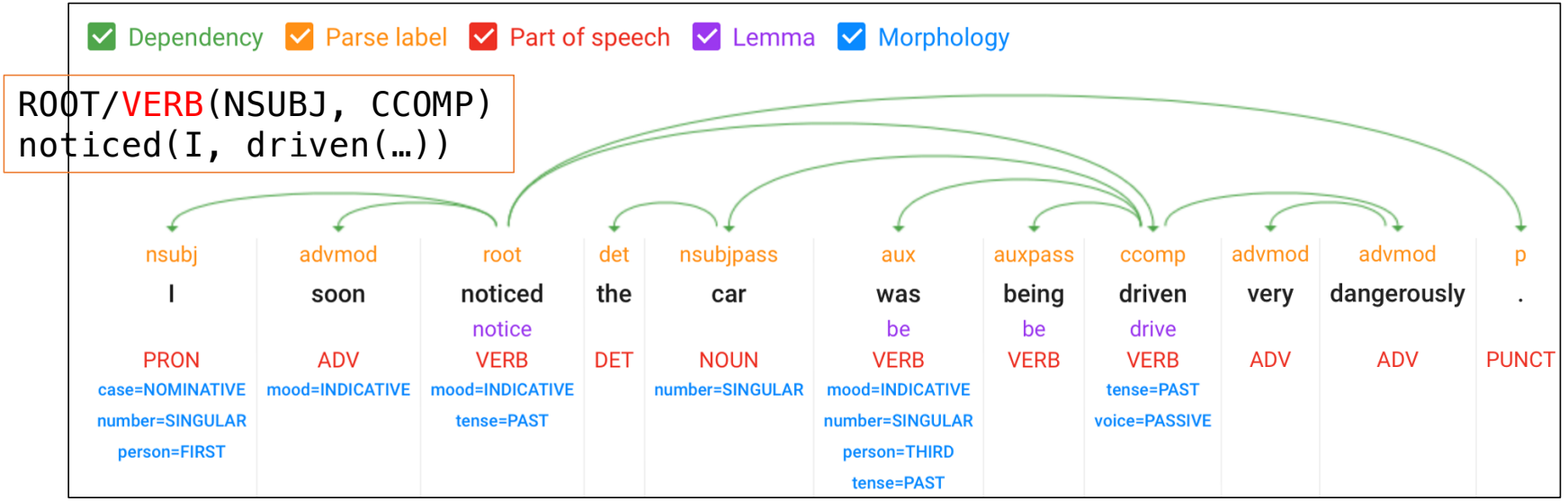
Background: Part 1

or scroll down until you get here ➠



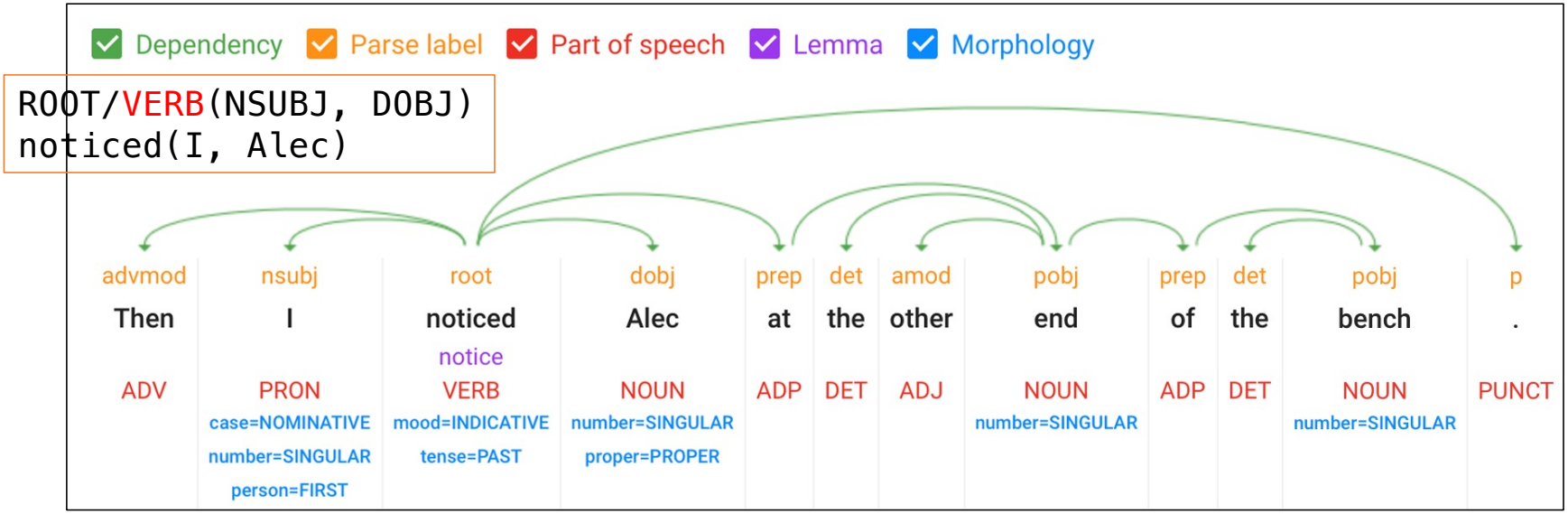
Background: Part 1

- 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: Part 1

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

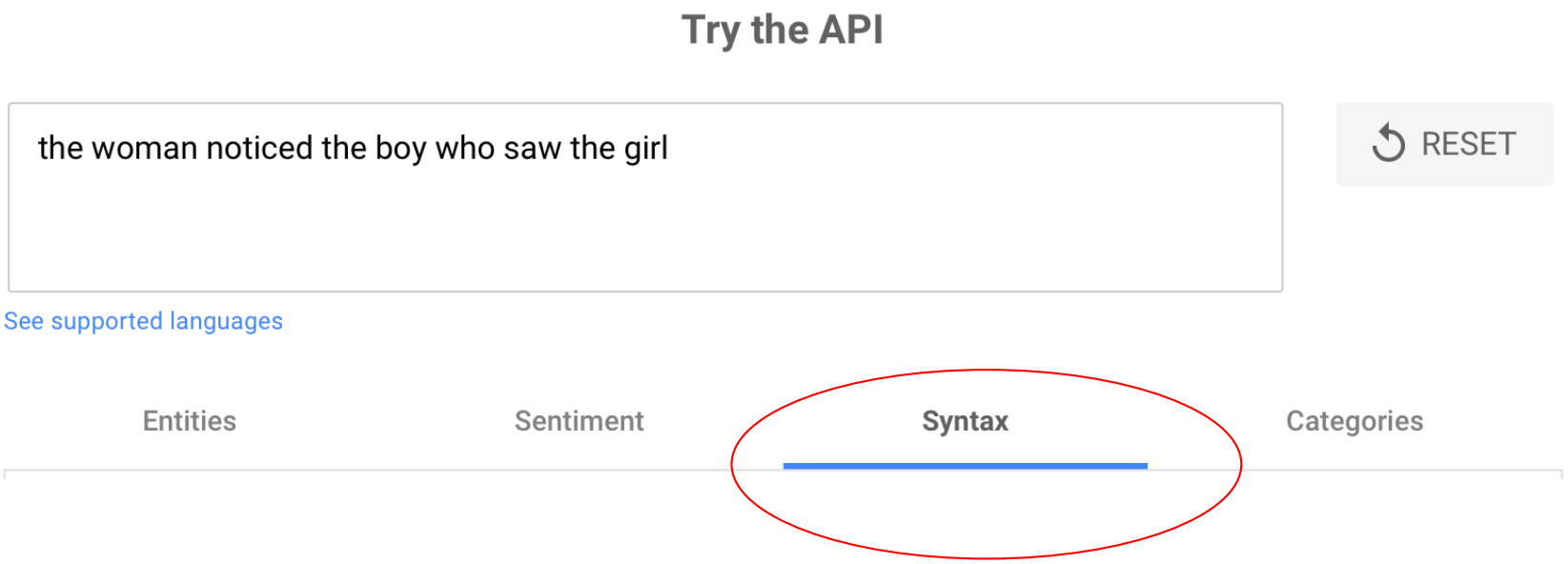


Google Natural Language

- 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).

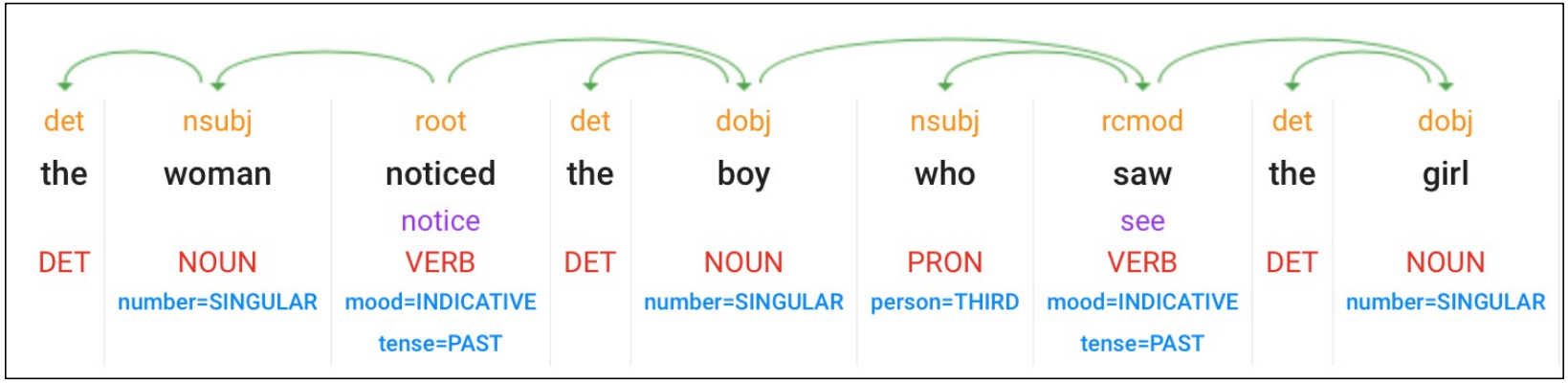
Background: Part 1

https://cloud.google.com/natural-language/



Background: Part 1

- `noticed(woman, boy)`
- `RCMOD` points back to `NOUN boy`
- `RCMOD/VERB(NSUBJ/PRON, DOBJ)`
- We infer `saw(boy, girl)`



Homework 9: Part 2

Question 1: We will compute predicate-argument structure using Perl regex with recursively embedded subject relative clauses:

1. the woman saw the boy
2. the woman saw the boy who saw the girl
3. the woman saw the boy who saw the girl who found the man
4. the woman saw the boy who saw the girl who found the man who chased the cat

- You can assume *the noun* for noun phrases (NPs) and *who* for the relative pronoun (*see next slide*)
- Write a Perl program using a regex to compute the predicate-argument relations and print them:

```
1. saw(woman, boy)
2. saw(woman, boy) saw(boy, girl)
3. saw(woman, boy) saw(boy, girl) found(girl, man)
4. saw(woman, boy) saw(boy, girl) found(girl, man) chased(man, cat)
```

Homework 9: Part 2

- Code should be general, i.e. you can swap out the verbs and common nouns etc., and it should still work.
- For simplicity, you may assume the patterns:
  - the $noun_1$ $verb$ the $noun_2$ ➠ $verb(noun_1, noun_2)$
  - the $noun_1$ who $verb$ the $noun_1$ ➠ $verb(noun_1, noun_2)$
- **Hints**:
  - note the pattern overlap, use lookahead `(?=pattern)`
  - you can collect the words together on the command line into a single string with `$sentence = qq/@ARGV/;`

Homework 9: Part 2

- Code should be general, i.e. you can swap out the verbs and common nouns etc., and it should still work.
- At the terminal, assume input/output will be something like:

```
$ perl hw9.perl the woman saw the boy
saw(woman, boy)
$ perl hw9.perl the woman saw the boy who saw the girl who found the man
saw(woman, boy)
saw(boy, girl)
found(girl, man)
$ perl hw9.perl the woman saw the boy who saw the girl who found the man who chased the cat
saw(woman, boy)
saw(boy, girl)
found(girl, man)
chased(man, cat)
$ perl hw9.perl the woman saw the boy who saw the girl who found the man who chased the cat who sensed the mouse
saw(woman, boy)
saw(boy, girl)
found(girl, man)
chased(man, cat)
sensed(cat, mouse)
$
```

Homework 9: Part 2

Question 2: a second type of embedded relative clauses.

Examples:

2. the woman sensed the boy the girl saw
3. the woman sensed the boy the girl the man found saw
4. the woman sensed the boy the girl the man the cat chased found saw

- Explain the differences between sentences 2–4 in Q2 vs. Q1 with respect to predicate-argument structure.

Homework 9: Part 2

Question 3: try Google Natural Language on the sentences with relative clauses from Q2.

2. the woman sensed the boy the girl saw.
3. the woman sensed the boy the girl the man found saw.
4. the woman sensed the boy the girl the man the cat chased found saw.

- Which one(s) does/do Google get wrong?
- As a human processor, which of 2–4 do you find very difficult to parse?

Homework 9: Part 2

- Extra Credit Question 4: based on what we've learnt so far, do you think it's possible to write a Perl regex program that prints the correct predicate-argument structure for the following examples from Q2?

Embedded relative clauses:

2. the woman sensed the boy the girl saw
3. the woman sensed the boy the girl the man found saw
4. the woman sensed the boy the girl the man the cat chased found saw