

LING/C SC 581:

Advanced Computational Linguistics

Lecture 16

Today's Topic

- Homework 7 Review
- You have already installed TREEBANK_3
- Stanford Tregex

Question 1 Review

- Modify parser options:

```
13 java -mx200m -cp "$scriptdir/*:" edu.stanford.nlp.parser.lexparser.Lexicalize...  
    dParser \t  
14 -outputFormat "penn,typedDependencies" -printPCFGkBest 5 edu/stanford/nlp/mo...  
    dels/lexparser/englishPCFG.ser.gz $*
```

dependencies in the usual way via the `-outputFormat` option, and each receives a score (log probability). The k best parses are extracted efficiently using the algorithm of Huang and Chiang (2005).

- Run it on the sentence:
 - Visiting relatives can be fun.
- Explain the two parses.

Question 1 Review

Parse 1 with score -45.7901119440794

```
(ROOT
  (S
    (S
      (VP (VBG Visiting)
        (NP (NNS
          relatives))))
      (VP (MD can)
        (VP (VB be)
          (ADJP (JJ fun))))
      (. .)))
```

csubj(fun-5, Visiting-1)

Parse 2 with score -45.91149179637432

```
(ROOT
  (S
    (NP (JJ Visiting) (NNS
      relatives))
    (VP (MD can)
      (VP (VB be)
        (ADJP (JJ fun))))
    (. .)))
```

nsubj(fun-5, relatives-2)

Question 2 Review

- What about the following ambiguous sentence?
 - *John saw the man with a telescope*
- Explain the PP attachment ambiguity.
 - [PP [P with_{instrument}] [NP a telescope]] attaches to VP
 - [PP [P with_{possession}] [NP a telescope]] attaches to NP

Question 2 Review

Parse 1 with score -44.61095468699932

```
(ROOT
  (S
    (NP (NNP John))
    (VP (VBD saw)
      (NP (DT the) (NN man))
      (PP (IN with)
        (NP (DT a) (NN
telescope))))
    (. .)))
```

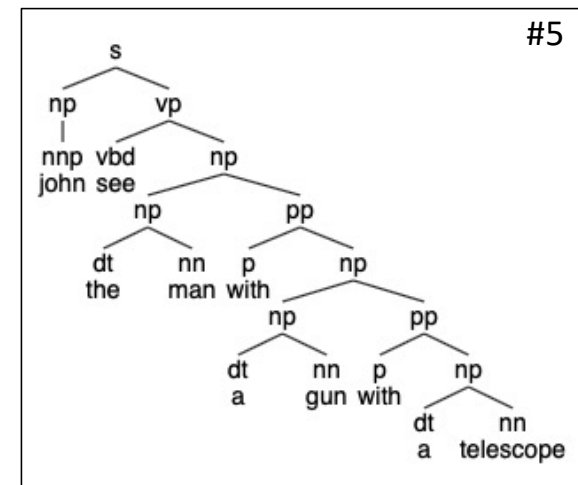
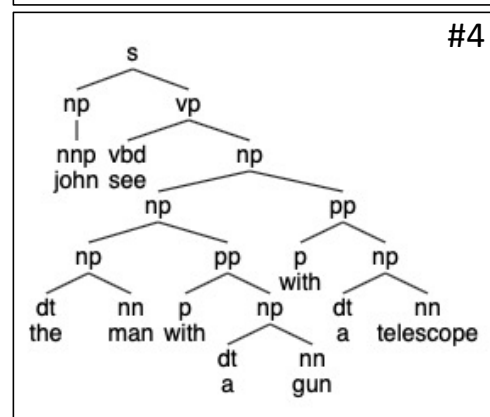
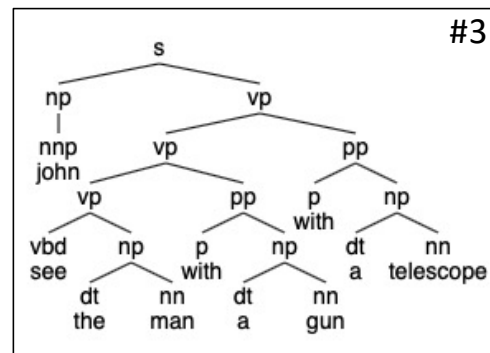
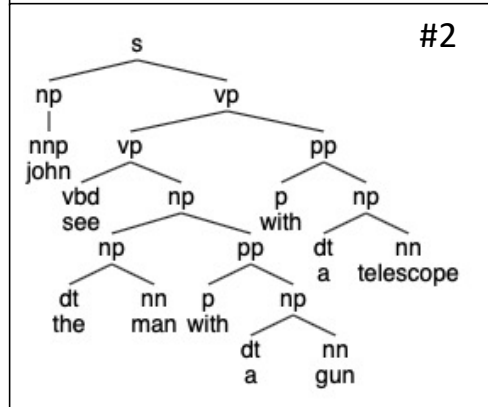
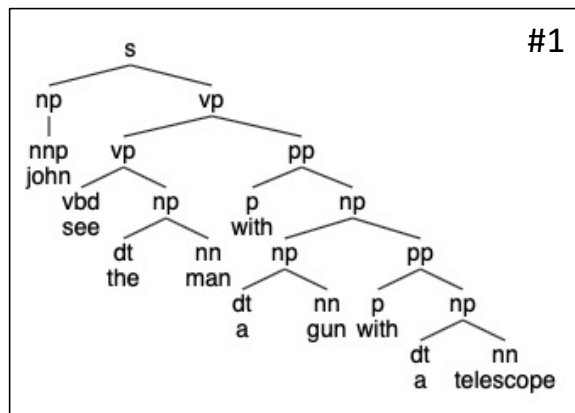
Parse 2 with score -45.487041898071766

```
(ROOT
  (S
    (NP (NNP John))
    (VP (VBD saw)
      (NP
        (NP (DT the) (NN man))
        (PP (IN with)
          (NP (DT a) (NN
telescope))))
      (. .)))
```

Question 3

- In principle, how many (syntactic) parses should there be for the following sentence?
 - John saw the man with a gun with a telescope
- Explain.
- Then modify `lexparser.sh` to give the required number of parses.
- Did it give the right outputs?
- The parses are ranked in order ($-\log\text{prob}$).
- Do you think the order is right?

Question 3 Review



Question 3 Review

Parse 1 with score -61.84777109324932

```
(ROOT #3
  (S
    (NP (NNP John))
    (VP (VBD saw)
      (NP (DT the) (NN man))
      (PP (IN with)
        (NP (DT a) (NN gun)))
      (PP (IN with)
        (NP (DT a) (NN telescope))))
    (. .)))
```

Parse 2 with score -62.176213689148426

```
(ROOT #1
  (S
    (NP (NNP John))
    (VP (VBD saw)
      (NP (DT the) (NN man))
      (PP (IN with)
        (NP
          (NP (DT a) (NN gun))
          (PP (IN with)
            (NP (DT a) (NN
telescope))))))
    (. .)))
```

Question 3 Review

Parse 3 with score -62.69277472048998

```
(ROOT
  (S
    (NP (NNP John))
    (VP (VBD saw)
      (NP
        (NP (DT the) (NN man))
        (PP (IN with)
          (NP (DT a) (NN gun))))
      (PP (IN with)
        (NP (DT a) (NN telescope))))
    (. .)))
```

#3

Parse 4 with score -63.32824368029833

```
(ROOT
  (S
    (NP (NNP John))
    (VP (VBD saw)
      (NP
        (NP (DT the) (NN man))
        (PP (IN with)
          (NP
            (NP (DT a) (NN gun))
            (PP (IN with)
              (NP (DT a) (NN
telescope))))))
            (. .)))
```

#5

Question 3 Review

Parse 5 with score -64.75296158343554

(ROOT

#4

(S

(NP (NNP John))

(VP (VBD saw)

(NP

(NP (DT the) (NN man))

(PP (IN with)

(NP (DT a) (NN gun)))


(PP (IN with)

(NP (DT a) (NN telescope))))

(. .)))

Install tregex

<https://nlp.stanford.edu/software/tregex.shtml>

The Stanford Natural Language Processing Group

people | publications | research blog | software | teaching | local

Software > Tregex, Tsurgeon and Semgrex

Tregex, Tsurgeon and Semgrex

[About](#) | [Questions](#) | [Mailing lists](#) | [Contents](#) | [Download](#) | [Extensions](#) | [Release history](#) | [FAQ](#)

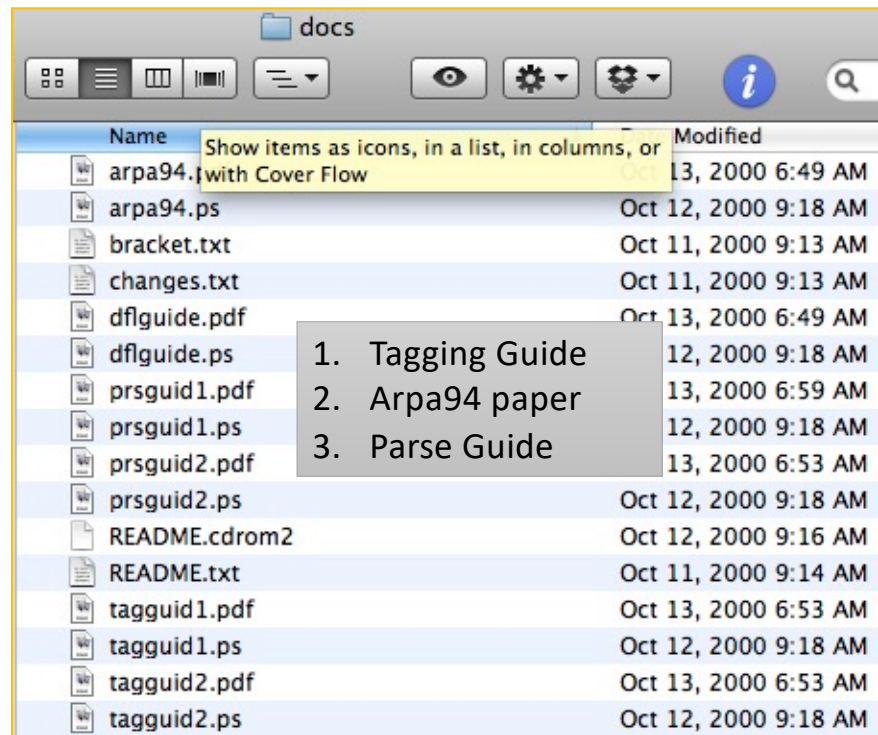
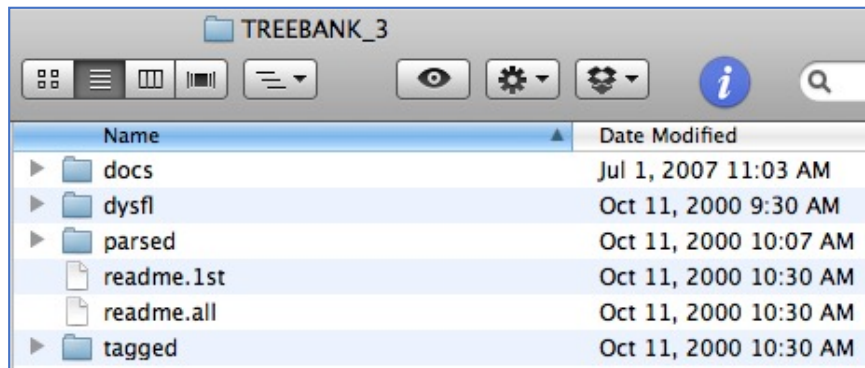
About

Tregex is a utility for matching patterns in trees, based on tree relationships and regular expression matches on nodes (the name is short for "tree regular expressions"). Tregex comes with **Tsurgeon**, a tree transformation language. Also included from version 2.0 on is a similar package which operates on dependency graphs (class `SemanticGraph`, called **semgrex**).

Tregex: The best introduction to Tregex is the brief powerpoint tutorial for [Tregex](#) by Galen Andrew. The best way to learn to use Tregex is by working with the GUI ([TregexGUI](#)). It has help screens which summarize the syntax of Tregex. You can find brief documentation of Tregex's pattern language on the [TregexPattern javadoc](#) page, and, of course, you should also be very familiar with [Java regular expression syntax](#). Tregex contains essentially the same functionality as [TGrep2](#) (which had a superset of the functionality of the original `tgrep`), plus several extremely useful relations for natural language trees, for example "A is the lexical head of B", and "A and B share a (hand-specified) variable substring" (useful for finding nodes coindexed with each other). Because it does not create preprocessed indexed corpus files, it is however somewhat slower than [TGrep2](#) when searching over large treebanks, but gains from being able to be run on any trees without requiring index construction. As a Java application, it is platform independent, and can be used programmatically in Java software. There is also both a graphical interface (also platform independent) and a command line interface through the [TregexPattern](#) main method. To launch the graphical interface double click the `stanford-tregex.jar` file.

Penn Treebank documentation

Find your TREEBANK_3 directory



tregex

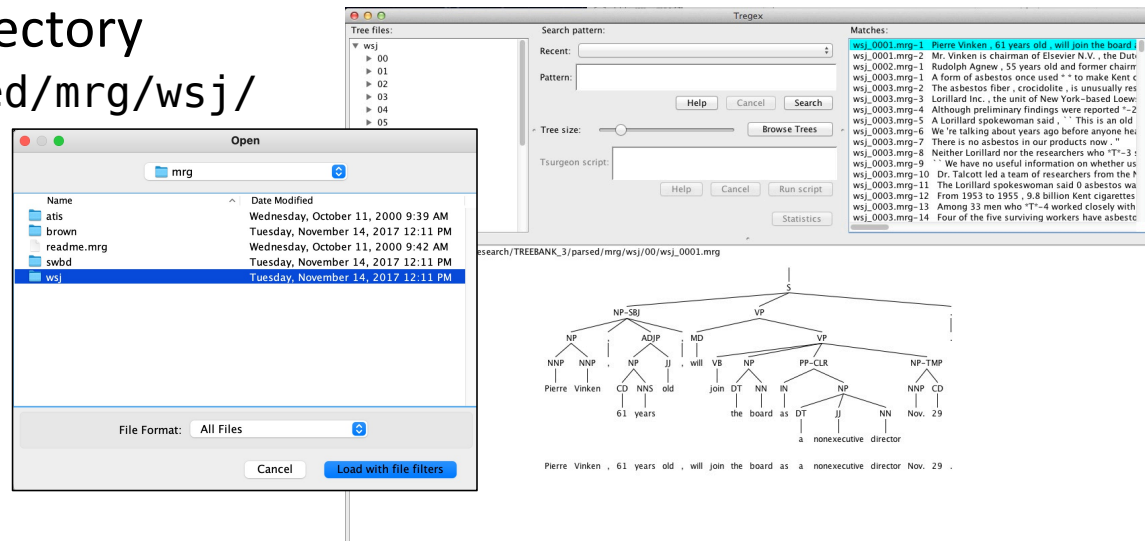
1. Shell file:

```
run-tregex-gui.command
#!/bin/sh
java -mx300m -cp `dirname $0`/stanford-tregex.jar edu.stanford.nlp.trees.tregex.gui.TregexGUI
```

2. Select the PTB directory

- TREEBANK_3/parsed/mrg/wsj/

3. Browse Trees



tregex

- Search

- NP-SBJ << (*dominates*) vs. < (*immediately dominates*) NNP

Statistics History		
Pattern	Trees Matched	Total Matches
NP-SBJ << NNP	19862	53523
NP-SBJ < NNP	11994	22740

Search pattern: NP-SBJ << NNP

Recent: NP-SBJ << NNP

Pattern: NP-SBJ << NNP

Tree size: [slider] Browse Trees

Tsurgeon script:

Match stats: 19862 unique trees found with 53523 total matches.

Statistics

Matches:

- wsj_0001.mrg-1 Pierre
- wsj_0001.mrg-2 Mr. Vin
- wsj_0003.mrg-1 A form
- wsj_0003.mrg-3 Lorilla
- wsj_0003.mrg-5 A Lon
- wsj_0003.mrg-8 Neithe
- wsj_0003.mrg-9 We
- wsj_0003.mrg-10 Dr. T
- wsj_0003.mrg-11 The U
- wsj_0003.mrg-16 The p
- wsj_0003.mrg-18 The f
- wsj_0003.mrg-19 The U
- wsj_0003.mrg-20 More
- wsj_0003.mrg-22 In Jul
- wsj_0003.mrg-28 The av
- wsj_0004.mrg-2 The av

EBANK_3/parsed/mrg/wsj/00/wsj_0001.mrg

Pattern: NP-SBJ < NNP

Tree size: [slider] Browse Trees

Tsurgeon script:

Match stats: 11994 unique trees found with 22740 total matches.

Statistics

BANK_3/parsed/mrg/wsj/00/wsj_0001.mrg

tregex

- README-tregex.txt

Tregex Pattern Syntax and Uses

Using a Tregex pattern, you can find only those trees that match the pattern you're looking for. The following table shows the symbols that are allowed in the pattern, and below there is more information about using these patterns.

Symbol	Meaning
A << B	A dominates B
A >> B	A is dominated by B
A < B	A immediately dominates B
A > B	A is immediately dominated by B
A \$ B	A is a sister of B (and not equal to B)
A .. B	A precedes B
A . B	A immediately precedes B
A ,, B	A follows B
A , B	A immediately follows B
A <<, B	B is a leftmost descendent of A
A <<- B	B is a rightmost descendent of A
A >>, B	A is a leftmost descendent of B
A >>- B	A is a rightmost descendent of B
A <, B	B is the first child of A
A >, B	A is the first child of B
A <- B	B is the last child of A
A >- B	A is the last child of B
A <# B	B is the last child of A
A ># B	A is the last child of B
A <i B	B is the ith child of A (i > 0)
A >i B	A is the ith child of B (i > 0)
A <-i B	B is the ith-to-last child of A (i > 0)
A >-i B	A is the ith-to-last child of B (i > 0)

A <: B	B is the only child of A
A >: B	A is the only child of B
A <<: B	A dominates B via an unbroken chain (length > 0) of unary local trees.
A >>: B	A is dominated by B via an unbroken chain (length > 0) of unary local trees.
A \$++ B	A is a left sister of B (same as \$.. for context-free trees)
A \$-- B	A is a right sister of B (same as \$., for context-free trees)
A \$+ B	A is the immediate left sister of B (same as \$. for context-free trees)
A \$- B	A is the immediate right sister of B (same as \$, for context-free trees)
A \$.. B	A is a sister of B and precedes B
A \$., B	A is a sister of B and follows B
A \$. B	A is a sister of B and immediately precedes B
A \$, B	A is a sister of B and immediately follows B
A <+(C) B	A dominates B via an unbroken chain of (zero or more) nodes matching description C
A >+(C) B	A is dominated by B via an unbroken chain of (zero or more) nodes matching description C
A .+(C) B	A precedes B via an unbroken chain of (zero or more) nodes matching description C
A ,+(C) B	A follows B via an unbroken chain of (zero or more) nodes matching description C
A <<# B	B is a head of phrase A
A >># B	A is a head of phrase B
A <# B	B is the immediate head of phrase A
A ># B	A is the immediate head of phrase B
A == B	A and B are the same node
A : B	[this is a pattern-segmenting operator that places no constraints on the relationship between A and B]

tregex

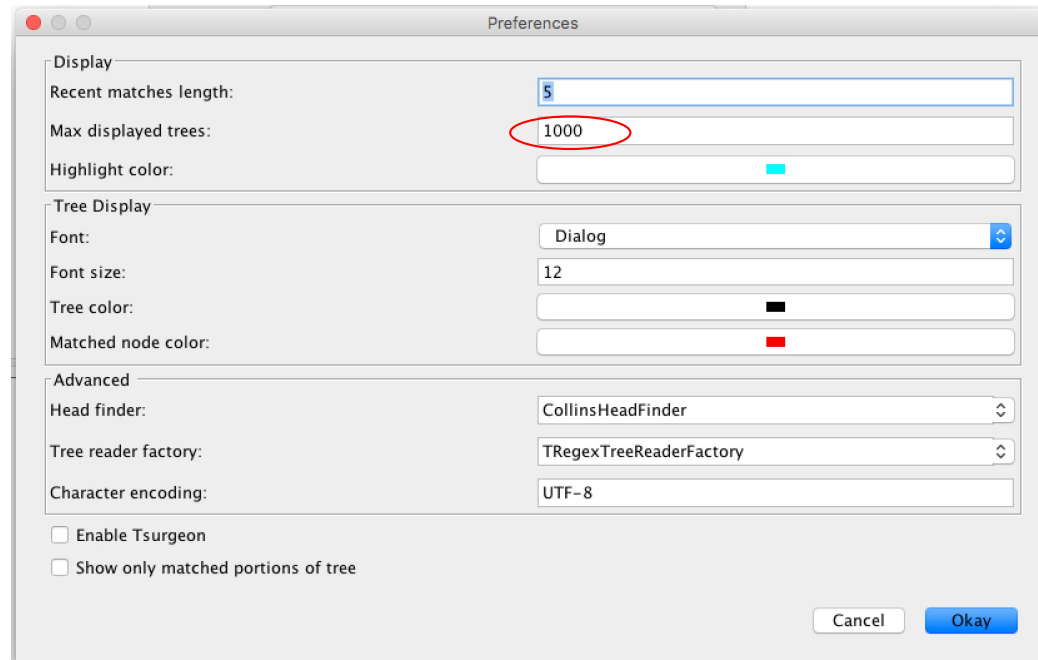
- The (best) introduction to **Tregex** is the brief powerpoint tutorial for **Tregex** by Galen Andrew.
 - https://nlp.stanford.edu/software/tregex/The_Wonderful_World_of_Tregex.ppt



The Wonderful World of
Tregex

tregex

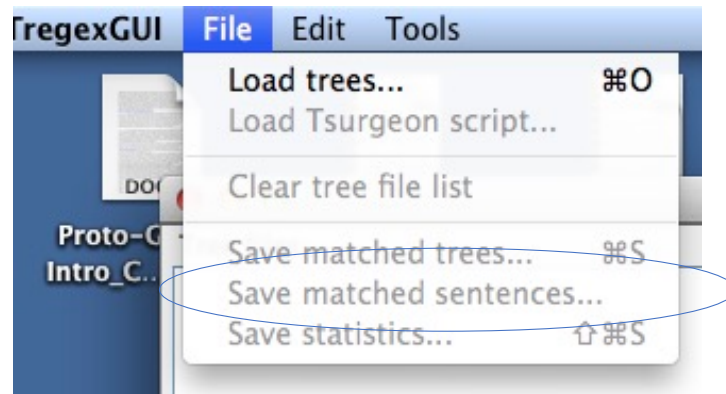
- Adjust Max displayed trees if needed:



tregex

- useful command line tool:
 - **diff <file1> <file2>**

```
dhcp-10-142-182-95:cleft searches sandiway$ diff whclf-0 whclf
4a5,6
> wsj_0415.mrg-5   Who that winner will be *T*-1 is highly uncertain .
> wsj_0415.mrg-22  `` And where we are *T*-1 is bad . ''
```



tregex

- Help: *tregex expression syntax is non-standard wrt bracketing*

Label descriptions can be literal strings, which much match labels exactly, or regular expressions in regular expression bars: `/regex/`. Literal string matching proceeds as String equality. In order to prevent ambiguity with other Tregex symbols, only standard "identifiers" are allowed as literals, i.e., strings matching `[a-zA-Z]([a-zA-Z0-9_])*`. If you want to use other symbols, you can do so by using a regular expression instead of a literal string. A disjunctive list of literal strings can be given separated by '|'. The special string '___' (two underscores) can be used to match any node. (WARNING!! Use of the '___' node description may seriously slow down search.) If a label description is preceeded by '@', the label will match any node whose *basicCategory* matches the description. NB: A single '@' thus scopes over a disjunction specified by '|': `@NP|VP` means things with basic category NP or VP. Label description regular expressions are matched as `find()`, as in Perl/tgrep; you need to specify `^` or `$` to constrain matches.

In a chain of relations, all relations are relative to the first node in the chain. For example, `(S < VP < NP)` means "an S over a VP and also over an NP". If instead what you want is an S above a VP above an NP, you should write `"S < (VP < NP)"`.

S	<	VP
S	<	NP

Nodes can be grouped using parens '(' and ')' as in `S < (NP $++ VP)` to match an S over an NP, where the NP has a VP as a right sister.

tregex

- Help: *tregex boolean syntax is also non-standard*

Boolean relational operators

Relations can be combined using the '&' and '|' operators, negated with the '!' operator, and made optional with the '?' operator. Thus `(NP < NN | < NNS)` will match an NP node dominating either an NN or an NNS. `(NP > S & $++ VP)` matches an NP that is both under an S and has a VP as a right sister.

Relations can be grouped using brackets '[' and ']'. So the expression

```
NP [< NN | < NNS] & > S
```

matches an NP that (1) dominates either an NN or an NNS, and (2) is under an S. Without brackets, & takes precedence over |, and equivalent operators are left-associative. Also note that & is the default combining operator if the operator is omitted in a chain of relations, so that the two patterns are equivalent:

```
(S < VP < NP)
(S < VP & < NP)
```

As another example, `(VP < VV | < NP % NP)` can be written explicitly as `(VP [< VV | [< NP & % NP]])`

Relations can be negated with the '!' operator, in which case the expression will match only if there is no node satisfying the relation. For example `(NP ! NNP)` matches only NPs not dominating an NNP. Label descriptions can also be negated with '!': `(NP !NNPINNS)` matches NPs dominating some node that is not an NNP or an NNS.

Relations can be made optional with the '?' operator. This way the expression will match even if the optional relation is not satisfied. This is useful when used together with node naming (see below).

tregex

- Help

Basic Categories

In order to consider only the "basic category" of a tree label, i.e. to ignore functional tags or other annotations on the label, prefix that node's description with the @ symbol. For example (@NP @/NN.?) This can only be used for individual nodes; if you want all nodes to use the basic category, it would be more efficient to use a {@link edu.stanford.nlp.trees.TreeNormalizer} to remove functional tags before passing the tree to the TregexPattern.

Segmenting patterns

The ":" operator allows you to segment a pattern into two pieces. This can simplify your pattern writing. For example, the pattern

S : NP

matches only those S nodes in trees that also have an NP node.

tregex

- $x <, y$, 1st child y ; $x <- y$, last child y ;
- $x \$+ y$, x immediate left sister of y

Naming nodes

Nodes can be given names (a.k.a. handles) using '='. A named node will be stored in a map that maps names to nodes so that if a match is found, the node corresponding to the named node can be extracted from the map. For example `(NP < NNP=name)` will match an NP dominating an NNP and after a match is found, the map can be queried with the name to retrieve the matched node using `TregexMatcher#getNode(Object o)` with (String) argument "name" (not "=name"). Note that you are not allowed to name a node that is under the scope of a negation operator (the semantics would be unclear, since you can't store a node that never gets matched to). Trying to do so will cause a `ParseException` to be thrown. Named nodes *can* be put within the scope of an optionality operator.

Named nodes that refer back to previous named nodes need not have a node description -- this is known as "backreferencing". In this case, the expression will match only when all instances of the same name get matched to the same tree node. For example: the pattern

```
(@NP <, (@NP $+ (/ / $+ (@NP $+ / /=comma))) <- =comma)
```

matches only an NP dominating exactly the sequence NP , NP , -- the mother NP cannot have any other daughters. Multiple backreferences are allowed. If the node w/ no node description does not refer to a previously named node, there will be no error, the expression simply will not match anything.

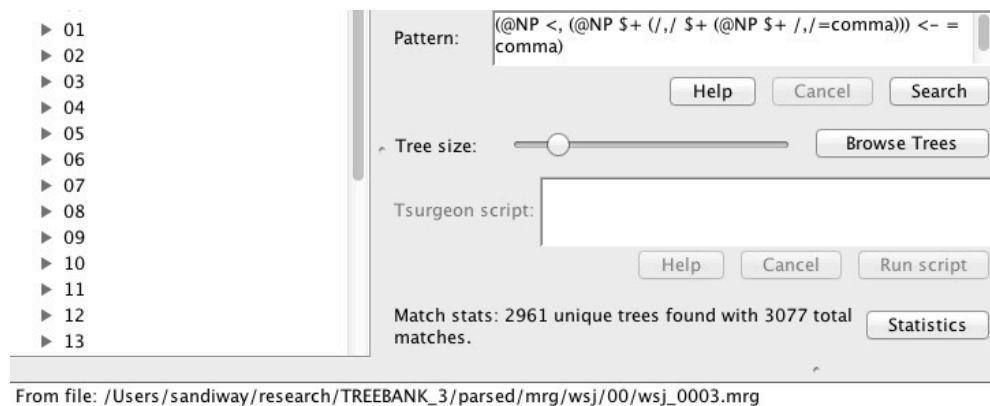
Another way to refer to previously named nodes is with the "link" symbol: '~'. A link is like a backreference, except that instead of having to be *equal to* the referred node, the current node only has to match the label of the referred to node. A link cannot have a node description, i.e. the '~' symbol must immediately follow a relation symbol.

tregex

- Pattern:

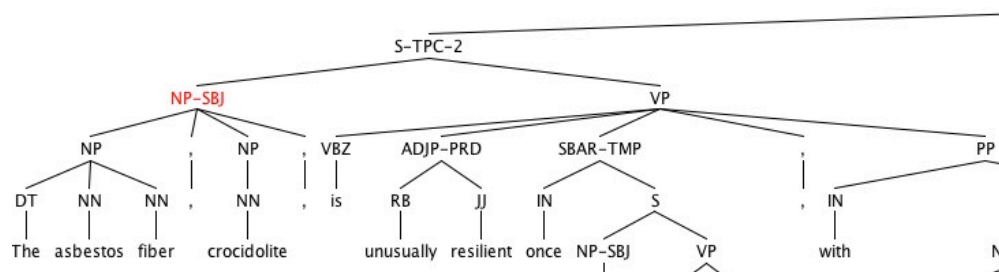
`(@NP <, (@NP $+ (/,/ $+ (@NP $+ /,/=comma))) <- =comma)`

same node



Key:

`<`, first child
`$+` immediate left sister
`<-` last child



tregex

- Help

Variable Groups

If you write a node description using a regular expression, you can assign its matching groups to variable names. If more than one node has a group assigned to the same variable name, then matching will only occur when all such groups capture the same string. This is useful for enforcing coindexation constraints. The syntax is

```
/ <regex-stuff> /#<group-number>%<variable-name>
```

For example, the pattern (designed for Penn Treebank trees)

```
@SBAR < /^WH.*-([0-9]+)$/#1%index << (__=empty < (/^-NONE-/ <  
/^\\*T\\*-*([0-9]+)$/#1%index)
```

will match only such that the WH- node under the SBAR is coindexed with the trace node that gets the name `empty`.

tregex

Pattern: `@SBAR < /^WH.*-([0-9]+)$/#1%index << (=_empty < (/^NONE-/ < /^T*-([0-9]+)$/#1%index))`

Tree size: Browse Trees

Tsurgeon script:

Match stats: 11898 unique trees found with 13906 total matches. [Statistics](#)

wsj_0003.mrg-8 Neither Lorillard nor the researchers
wsj_0003.mrg-13 Among 33 men who *T*-4 worked c
wsj_0003.mrg-16 `` The morbidity rate is a striking fi
wsj_0003.mrg-18 The plant , which *T*-1 is owned *-
wsj_0003.mrg-19 The finding probably will support th
wsj_0003.mrg-20 The U.S. is one of the few industriali
wsj_0003.mrg-24 About 160 workers at a factory that
wsj_0003.mrg-25 Areas of the factory "ICH"-2 were p
wsj_0003.mrg-27 Workers described `` clouds of blue
wsj_0004.mrg-15 It invests heavily in dollar-denomin
wsj_0005.mrg-1 J.P. Bolduc , vice chairman of W.R. Gra
wsj_0005.mrg-2 He succeeds Terrence D. Daniels , for
wsj_0008.mrg-4 Legislation 0 *T*-1 to lift the debt ce
wsj_0010.mrg-1 When it 's time for their biannual pow
wsj_0010.mrg-5 The idea , of course : * to prove to 12

Tree structure diagram:

```
graph TD
    S1[S] --- NP_SBJ1[NP-SBJ]
    S1 --- VP1[VP]
    NP_SBJ1 --- PRP1[PRP]
    PRP1 --- it1[it]
    VP1 --- VBZ1[VBZ]
    VBZ1 --- enters1[enters]
    VP1 --- NP2[NP]
    NP2 --- DT1[DT]
    DT1 --- the1[the]
    NP2 --- NNS1[NNS]
    NNS1 --- lungs1[lungs]

    IN[IN] --- with[with]

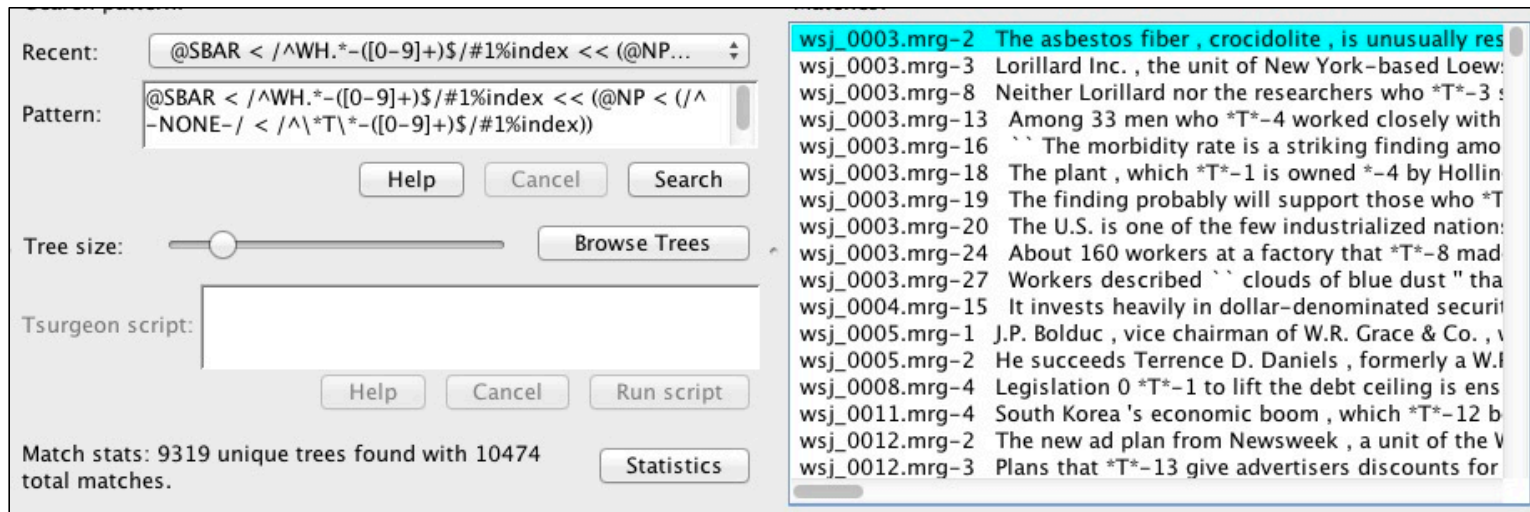
    S2[S-NOM] --- NP_SBJ2[NP-SBJ]
    S2 --- VP2[VP]
    NP_SBJ2 --- NP3[NP]
    NP3 --- RB1[RB]
    RB1 --- even1[even]
    NP3 --- JJ1[JJ]
    JJ1 --- brief1[brief]
    NP3 --- NNS2[NNS]
    NNS2 --- exposures1[exposures]
    NP_SBJ2 --- PP1[PP]
    PP1 --- TO1[TO]
    TO1 --- to1[to]
    PP1 --- NP4[NP]
    NP4 --- PRP2[PRP]
    PRP2 --- it2[it]

    VP2 --- VBG1[VBG]
    VBG1 --- causing1[causing]
    VP2 --- NP5[NP]
    NP5 --- NNS3[NNS]
    NNS3 --- symptoms1[symptoms]
    NP5 --- WHNP1[WHNP-1]
    WHNP1 --- that1[that]
    NP5 --- S3[S]
    S3 --- NP_SBJ3[NP-SBJ]
    NP_SBJ3 --- NONE1[-NONE-]
    NONE1 --- T1[*T*-1]
    S3 --- VP3[VP]
    VP3 --- VBP1[VBP]
    VBP1 --- show1[show]
    VP3 --- PRT1[PRT]
    PRT1 --- up1[up]
    VP3 --- ADVP_TMP1[ADVP-TMP]
    ADVP_TMP1 --- NP6[NP]
    NP6 --- NNS4[NNS]
    NNS4 --- decades1[decades]
    ADVP_TMP1 --- JJ1[JJ]
    JJ1 --- later1[later]
```

tregex

- Different results from:

- `@SBAR < /^WH.*-([0-9]+)$/#1%index << (@NP < (/^NONE-/ < /^*T*-([0-9]+)$/#1%index))`



tregex

Pattern: @SBAR < / ^WH.*-([0-9]+)\$/#1%index << _ < (/ ^-N ONE- / < / ^*T*-([0-9]+)\$/#1%index))

Tree size: Browse Trees

Tsurgeon script: Run script

Match stats: 11898 unique trees found with 13906 total matches. Statistics

wsj_0003.mrg-8 Neither Lorillard n
wsj_0003.mrg-13 Among 33 men v
wsj_0003.mrg-16 The morbidity
wsj_0003.mrg-18 The plant , which
wsj_0003.mrg-19 The finding prob
wsj_0003.mrg-20 The U.S. is one o
wsj_0003.mrg-24 About 160 work
wsj_0003.mrg-25 Areas of the fact
wsj_0003.mrg-27 Workers describ
wsj_0004.mrg-15 It invests heavil
wsj_0005.mrg-1 J.P. Bolduc , vice c
wsj_0005.mrg-2 He succeeds Terre
wsj_0008.mrg-4 Legislation 0 *T*-
wsj_0010.mrg-1 When it 's time for
wsj_0010.mrg-5 The idea , of cour

ANK_3/parsed/mrg/wsj/00/wsj_0003.mrg

```
graph TD
    S --> NP_SBJ[NP-SBJ]
    S --> VP1[VP]
    S --> SBAR_2[SBAR-2]
    NP_SBJ --> PP[PP]
    NP_SBJ --> NP1[NP]
    PP --> DT1[DT]
    PP --> NN1[NN]
    DT1 --> the1[the]
    NN1 --> factory[factory]
    NP1 --> NONE1[-NONE-]
    NONE1 --> ICH1[*ICH*-2]
    VP1 --> VBD1[VBD]
    VBD1 --> were[were]
    VP1 --> ADJP_PRD[ADJP-PRD]
    ADJP_PRD --> RB[RB]
    ADJP_PRD --> JJ[JJ]
    RB --> particularly[particularly]
    JJ --> dusty[dusty]
    VP1 --> WHADVP_1[WHADVP-1]
    WHADVP_1 --> WRB[WRB]
    WRB --> where[where]
    SBAR_2 --> S2[S]
    S2 --> NP_SBJ_8[NP-SBJ-8]
    S2 --> VP2[VP]
    NP_SBJ_8 --> DT2[DT]
    NP_SBJ_8 --> NN2[NN]
    DT2 --> the2[the]
    NN2 --> crocidolite[crocidolite]
    VP2 --> VBD2[VBD]
    VBD2 --> was[was]
    VP2 --> VBN[VBN]
    VBN --> used[used]
    VP2 --> NP3[NP]
    NP3 --> NONE2[-NONE-]
    NONE2 --> T8[*T*-8]
    VP2 --> ADVP_LOC[ADVP-LOC]
    ADVP_LOC --> NONE3[-NONE-]
    NONE3 --> T1[*T*-1]
```

Reason for difference

Example:

WHADVP also possible (not just WHNP)