Stanford University

S

NLP

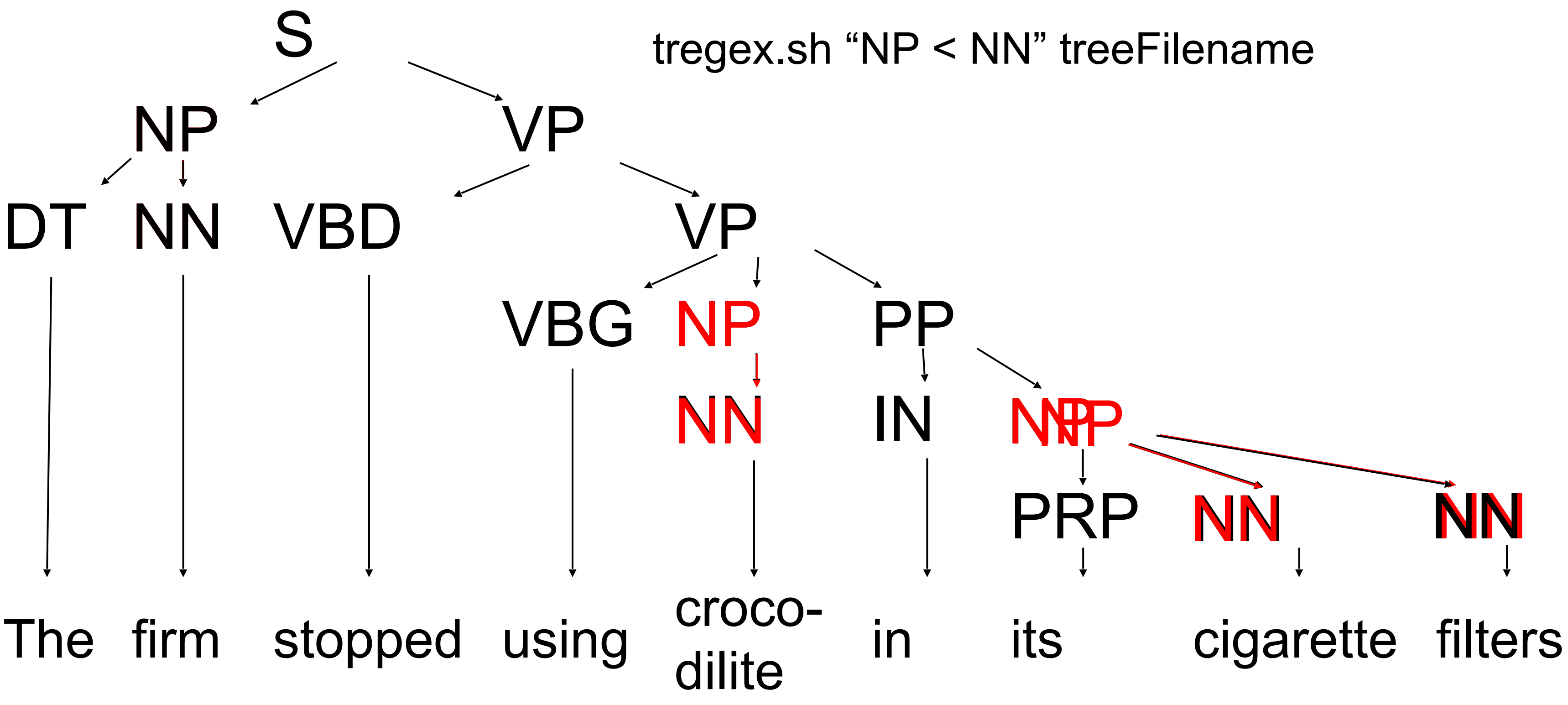
Natural Language Processing

The Wonderful World of Tregex

What is Tregex?

- A java program for identifying patterns in trees
- Like regular expressions for strings, based on tgrep syntax
- Simple example: NP < NN

```
tregex.sh "NP < NN" treeFilename
```



S
NP VP
DT NN VBD VP
VBG NP PP
NN IN NP
PRP NN NN
The firm stopped using croco-dilite in its cigarette filters

Syntax (Node Descriptions)

- The basic units of Tregex are Node Descriptions
- Descriptions match node labels of a tree
  - Literal string to match: NP
    - Disjunction of literal strings separated by ‘|’: NP|PP|VP
  - Regular Expression (Java 5 regex): /NN.?/
    - Matches NN, NNP, NNS
  - Wildcard symbol: __ (two underscores)
    - Matches any node (warning: can be SLOW!)
- Descriptions can be negated with !: !NP
- Preceding desc with @ uses basic category
  - @NP will match NP-SBJ

Syntax (Relations)

- Relationships between tree nodes can be specified
- There are many different relations. Here are a few:

<table>
<tr><th>Symbol</th><th>Description</th><th>Symbol</th><th>Description</th></tr>
<tr><td>A < B</td><td>A is the parent of B</td><td>A << B</td><td>A is an ancestor of B</td></tr>
<tr><td>A $ B</td><td>A and B are sisters</td><td>A $+ B</td><td>B is next sister of A</td></tr>
<tr><td>A <i B</td><td>B is i^{th} child of A</td><td>A <: B</td><td>B is only child of A</td></tr>
<tr><td>A <<# B</td><td>A on head path of B</td><td>A <<- B</td><td>B is rightmost descendent</td></tr>
<tr><td>A .. B</td><td colspan="3">A precedes B in depth-first traversal of tree</td></tr>
<tr><td>A <+(C) B</td><td colspan="3">A dominates B via unbroken chain of Cs</td></tr>
</table>

Building complex expressions

- Relations can be strung together for “and”
  - All relations are relative to first node in string
  - NP < NN $ VP
    - “An NP over an NN and w/ sister VP”
  - & symbol is optional: NP < NN & $ VP
- Nodes can be grouped w/ parentheses
  - NP < (NN < dog)
    - “An NP over an NN that is over ‘dog’ ”
  - Not the same as NP < NN < dog
- Ex: NP < (NN < dog) $ (VP <<# (barks > VBZ))
  - “An NP both over an NN over ‘dog’ and with a sister VP headed by ‘barks’ under VBZ”

Other Operators on Relations

- Operators can be combined via “or” with |
  - Ex: NP < NN | < NNS
  - “An NP over NN or over NNS”
- By default, & takes precedence over |
  - Ex: NP < NNS | < NN & $ VP
  - “NP over NNS OR both over NN and w/ sister VP”
  - Equivalent operators are left-associative
- Any relation can be negated with “!” prefix
  - Ex: NP !<< NNP
  - “An NP that does not dominate NNP”

Grouping relations

- To specify operation order, use [and]
  - Ex: NP [< NNS | < NN] $ VP
  - “An NP either over NNS or NN, and w/ sister VP”
- Grouped relations can be negated
  - Just put ! before the [
- Already we can build very complex expressions!
  - NP <- /NN.?/ > (PP <<# (IN ![< of | < on]))
  - “An NP with rightmost child matching /NN.?/ under a PP headed by some preposition (IN) that is not either ‘of’ or ‘on’ ”

Named Nodes

- Sometimes we want to find which nodes matched particular sub-expressions
  - Ex: /NN.?/ $- @JJ|DT
  - What was the modifier that preceded the noun?
- Name nodes with = and if expression matches, we can retrieve matching sub-expr with name
  - Ex: /NN.?/ $- @JJ|DT=premod
  - Subtree with root matching @JJ|DT is stored in a map under key “premod”
- Note:
  - named nodes are not allowed in scope of negation

Optional Nodes

- Sometimes we want to try to match a sub-expression to retrieve named nodes if they exist, but still match root if sub-expression fails.
- Use the optional relation prefix '?'
- Ex: NP < (NN ?$- JJ=premod) $+ CC $++ NP
  - Matches NP over NN with sisters CC and NP
  - If NN is preceded by JJ, we can retrieve the JJ using the key "premod"
  - If there is no JJ, the expression will still match
- Cannot be combined with negation

Use of the tregex GUI application

- Double-click the Stanford Tregex application (Mac OS X) or run-tregex-gui (Windows/Linux)
  - Equivalent to running:
    - java -mx300m –cp stanford-tregex.jar edu.stanford.nlp.trees.tregex.gui.TregexGUI
- Set preferences if necessary (e.g., for non-English treebanks)
- Load trees from File menu
- Enter a search pattern in the Pattern box
- Click Search
  - Also try the useful Help button

Use of tregex from the command-line

- tregex.sh “pattern” filename
- tregex.bat “pattern” filename
  - Equivalent to:
    - java -cp 'stanford-tregex.jar:' edu.stanford.nlp.trees.tregex.TregexPattern “pattern” filename
  - The pattern almost always needs to be quoted because of special characters it contains like < and >
  - If the filename is a directory, all files under it are searched

Command-line options

- Place any of these before the pattern:
  - -C only count matches, don’t print
  - -w print whole matching tree, not just matching subtree
  - -f print filename
  - -i <filename> read search pattern from <filename> rather than the command line
  - -s print each match on one line, instead of multi-line pretty- printing
  - -u only print labels of matching nodes, not complete subtrees !! -t print terminals only

Use of Tregex Java classes

- Tregex usage is like `java.util.regex`

```
String s = “@NP $+ (CC=conj $+ (@NP <- /^PP/))”;
TregexPattern p = TregexPattern.compile(s);
TregexMatcher m = p.matcher(tree);
while (m.find()) {
   m.getMatch().pennPrint();
}
```

- Named nodes are retrieved with `getNode()`

```
while (m.find()) {
   Tree conjSubTree = m.getNode(“conj”);
   System.out.println(conjSubTree.value());
}
```

Options

- TregexPatterns use a HeadFinder for <<# and BasicCategory map for @
- BasicCategory map is Function from String ➔ String
- Defaults are for English Penn Treebank
- To change these, use TregexPatternCompiler

```
HeadFinder hf = new ChineseHeadFinder();
TreebankLanguagePack chineseTLP =
   new ChineseTreebankLanguagePack();
Function bcf = chineseTLP.getBasicCategoryFunction();
TregexPatternCompiler c =
   new TregexPatternCompiler(hf, bcf);
TregexPattern p = c.compile(s);
```

Tregex (and Tsurgeon)

- Available for download at:
  - http://nlp.stanford.edu/software/tregex.shtml
- Tregex and Tsurgeon were initially written by Galen Andrew and Roger Levy
  - Roger Levy and Galen Andrew. 2006. Tregex and Tsurgeon: tools for querying and manipulating tree data structures. *Proceedings of LREC 2006*.
    - http://nlp.stanford.edu/pubs/levy_andrew_lrec2006.pdf
- Formats handled by the tool:
  - Penn Treebank
  - Others if you provide Java TreeReader's
    - E.g., it has been used with CCG

ENJOY!!!

Stanford University

S

NLP

Natural Language Processing

The Wonderful World of Tregex