Aston Corpus Summer School
Carola Trips *08/02/2011*

CorpusSearch: Search tool for parsed corpora

(http://corpussearch.sourceforge.net/CS-manual/WhatIs.html)

What is CorpusSearch?

CorpusSearch finds linguistic structures in a corpus of parsed, labelled sentences. It also has other features, including support for the automatic creation of coding strings for statistical analysis and the automatic creation of a lexicon for a corpus.

input to CorpusSearch

CorpusSearch needs two pieces of information:

- a corpus of sentences to search (source file(s)).
- a specification of what structures to search for (command file).

source file(s)

A source file is any file that contains parsed, labelled sentences. This could be a file from the Penn Parsed Corpora of Historical English or from another parsed corpus. It could also be an output file from a previous search, or perhaps a file of sentences that the user has cut and pasted together. Any number of source files can be searched in a single one run of CorpusSearch.

command file

The command file contains a query, which describes the structures being searched for, and possibly additional control and output specifications. This additional material may specify the node boundaries within which to search, and may choose various options for specifying the form of the output.

output of CorpusSearch

CorpusSearch always builds a text output file, containing the sentences with the specified structure, and basic statistics.

search output

The output file contains the sentences that were found to contain the searched-for structure, along with comments describing where the structures were found. Statistics are kept detailing the number of "hits," that is, distinct constituents containing the structure, the number of matrix sentences ("tokens") containing hits, and the total number of tokens in the file. Notice that the number of hits may change depending on the definition of the boundary node.

about the query language

The CorpusSearch query language has these basic components:

- **search-function calls**. Each search function looks for one basic relationship, like "dominates" or "precedes".
- **arguments to search-function calls**. These describe the nodes being searched for. Search function arguments may take the form of an or-list, may include wild cards, and may be negated.

- AND, OR and NOT. AND, OR, and NOT are used as in basic formal logic.
- open parenthesis, "(", and close parenthesis, ")". Parentheses are used as in basic formal logic.

search function calls

The most basic query is a single search-function call. For instance, here is a query that searches for nodes labelled QP ("quantifier phrase") that immediately dominate nodes labelled CONJ ("co-ordinating conjunction"):

```
(QP iDominates CONJ)
```

and here is a sentence found by the query:

```
/~*
and so he is bo+te more and lasse to his seruaunt.
(CMWYCSER,351.2223)
*~/

/*
      1 IP-MAT: 9 QP, 10 CONJ bo+te
      1 IP-MAT: 9 QP, 12 CONJ and
*/

(0
   (1 IP-MAT (2 CONJ and)
             (3 ADVP (4 ADV so))
             (5 NP-SBJ (6 PRO he))
             (7 BEP is)
             (8 ADJP
                     (9 QP (10 CONJ bo+te) (11 QR more) (12 CONJ and) (13 QR
                      lasse))
                     (14 PP (15 P to)
                            (16 NP (17 PRO$ his) (18 N seruaunt))))
             (19 E_S .))
      (ID CMWYCSER,351.2223))
```

Any number of search-function calls may be combined into more complex queries using AND, OR, and NOT.

wild cards and escaping wild cards

CorpusSearch supports two wild cards, namely * and #.

*****

* works as in regular expressions, that is, it stands for any string of symbols. For instance, "CP*" means any label beginning with the letters CP (e.g. CP, CP-ADV, CP-QUE-SPE). "*-SPE" means any label ending with "-SPE", and *hersum* means any string containing the substring "hersum" (e.g., "hersumnesse", "unhersumnesse"). * by itself will match any string. * may be used anywhere in the function argument; beginning, middle or end.

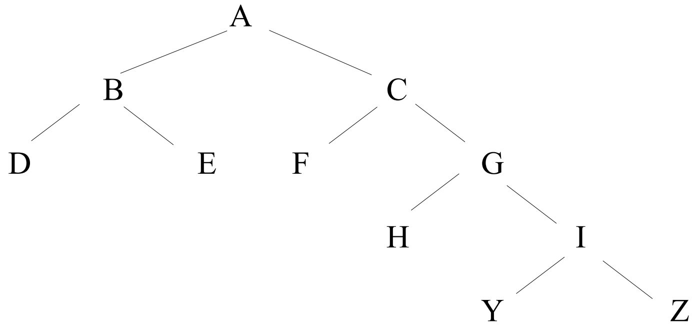
Search functions

The relations of Dominance and Precedence are important here:

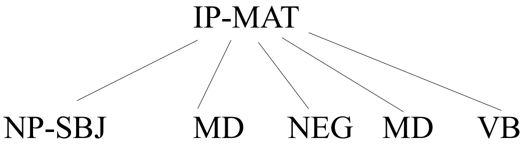
Dominance: Node A dominates node B if and only if A is higher up in the tree than B and if you can trace a line from A to B going only downwards.

Precedence: Node A precedes node B if and only if A is to the left of B and neither A dominates B nor B dominates A.
(definitions from Haegeman 1994: 85)

(a)

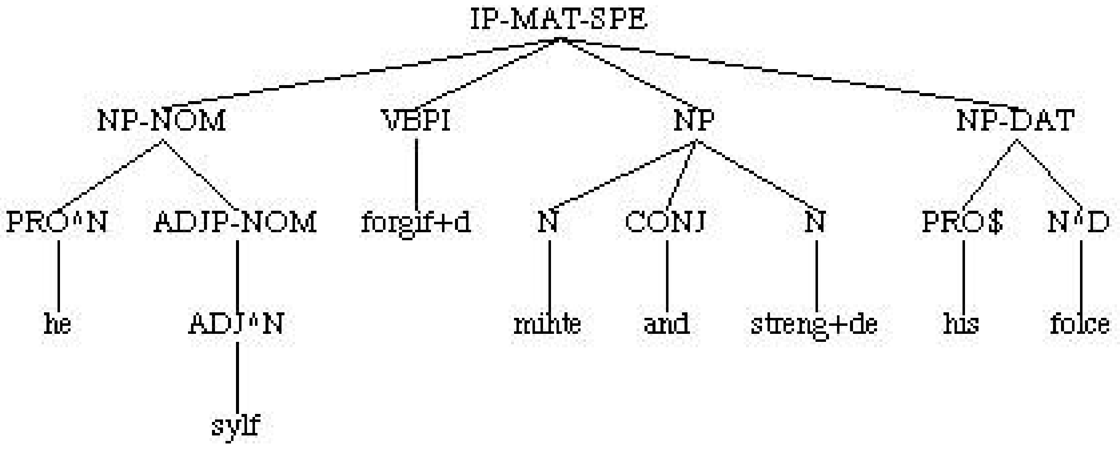


(b)



(c)

```
( (IP-MAT-SPE (NP-NOM (PRO^N he)
                      (ADJP-NOM (ADJ^N sylf)))
              (VBPI forgif+d)
              (NP (N mihte) (CONJ and) (N streng+de))
              (NP-DAT (PRO$ his) (N^D folce))
              (. ;)) (ID copreflives,+ALS_[Pref]:19.11))
```



Exists (variants: exists)
exists searches for label or text anywhere in the sentence. For instance, this query:

```
(MD0 exists)
```

will find this sentence:

```
/~*
but I fere me that I shal not conne wel goo thyder /
(CMREYNAR,14.261)
*~/

/*
    10 IP-SUB: 15 MD0 conne
*/

((10 IP-SUB
               (11 NP-SBJ (12 PRO I))
               (13 MD shal)
               (14 NEG not)
               (15 MD0 conne)
               (16 ADVP (17 ADV wel))
               (18 VB goo)
               (19 ADVP-DIR (20 ADV thyder)))
      (ID CMREYNAR,14.261))
```

Dominates (variants: dominates, Doms, doms)
dominates means "dominates to any generation." That is, y is contained in the sub-tree dominated by x.

So this query:

```
node: (IP-SUB dominates NP-SBJ)
```

will find this sentence:

```
that I shal not conne wel goo thyder /
(CMREYNAR,14.261)
*~/

(
      (10 IP-SUB
                 (11 NP-SBJ (12 PRO I))
                 (13 MD shal)
                 (14 NEG not)
                 (15 MD0 conne)
                 (16 ADVP (17 ADV wel))
                 (18 VB goo)
                 (19 ADVP-DIR (20 ADV thyder)))
      (ID CMREYNAR,14.261))
```

DomsWords (variants: domsWords, domswords)
domsWords counts the number of words dominated by the search-function argument. So "domsWords 4" means "dominates 4 words", domsWords 2 mean "dominates 2 words", and so on.

So this query:

```
node: NP*
(NP-OB* domsWords 3)
```

will return this structure:

```
/~*
and by kynge Ban and Bors his counceile they lette brenne and
destroy all the
contrey before them there they sholde ryde.
(CMMALORY,20.613)
*~/


    24 NP-OB1: 27 N contrey
*/

(      (24 NP-OB1 (25 Q all)
                 (26 D the)
                 (27 N contrey)
                 (28 CP-REL *ICH*-1))
      (ID CMMALORY,20.613))
```

HasSister (variants: hasSister, hassister)
x hasSister y if x and y have the same mother. It doesn't matter whether x precedes y or y precedes x. So this query:

```
node: IP*
query: (NP-SBJ hasSister BE*)
```

finds the sentence:

```
/~*
indeede I must be gone:
(DELONEY,69.13)
*~/
/*
1 IP-MAT-SPE:  5 NP-SBJ, 10 BE
*/

( (IP-MAT-SPE (PP (P+N indeede))
              (NP-SBJ (PRO I))
              (MD must)
              (BE be)
              (VBN gone)
              (. :))
  (ID DELONEY,69.13))
```

iDominates (variants: idominates, iDoms, idoms)

iDominates means "immediately dominates". That is, x dominates y if y is a child (exactly one generation apart) of x. So this query:

```
((NP* iDominates FP) AND (FP iDominates ane))
```

finds this sentence:

```
/~*
Sythen he ledes +tam by +tar ane,
(CMROLLEP,118.978)
*~/

/*
    1 IP-MAT: 11 NP, 13 FP ane
*/

(0
   (1 IP-MAT
           (2 ADVP-TMP (3 ADV Sythen))
           (4 NP-SBJ (5 PRO he))
           (6 VBP ledes)
           (7 NP-OB1 (8 PRO +tam))
           (9 PP (10 P by)
                 (11 NP (12 PRO$ +tar) (13 FP ane)))
           (14 E_S ,))
      (ID CMROLLEP,118.978))

/*
```

Notice that "iDominates" also describes the relationship between a label and its associated text (e.g., "FP" and "ane").

Precedes (variants: precedes, Pres, pres)
"x precedes y" means "x comes before y in the tree but x does not dominate y". So this query:

```
(VB precedes NP-OB*)
```

produces this output:

```
/~*
thenne have ye cause to make myghty werre upon hym. '
(CMMALORY,2.25)

    9 IP-INF-PRP: 11 VB make, 12 NP-OB1
*/

(
      (9 IP-INF-PRP (10 TO to)
                    (11 VB make)
                    (12 NP-OB1 (13 ADJ myghty)
                               (14 N werre)
                               (15 PP (16 P upon)
                (17 NP (18 PRO hym)))))
      (ID CMMALORY,2.25))
```

iPrecedes (variants: iprecedes, iPres, ipres)
This function is true if and only if its first argument immediately precedes its second argument in the text string spanned by the parse tree.

The following query:

```
query: ([1]as iPrecedes sone) AND (sone iPrecedes [2]as)
```

produces this output:

```
/~*
and as sone as he myght he toke his horse
(CMMALORY,206.3401)
*~/
/*
1 IP-MAT:  6 as, 8 sone, 11 as
*/

( (IP-MAT (CONJ and)
          (ADVP-TMP (ADVR as)
                    (ADV sone)
                    (PP (P as)
                        (CP-CMP (WADVP-1 0)
                                (C 0)
                                (IP-SUB (ADVP-TMP *T*-1)
                                        (NP-SBJ (PRO he))
                                        (MD myght)
                                        (VB *)))))
          (NP-SBJ (PRO he))
          (VBD toke)
          (NP-OB1 (PRO$ his) (N horse)))
  (ID CMMALORY,206.3401))
```