CorpusSearch Workshop

Beatrice Santorini
University of Pennsylvania

15 August 2007
Overview

- Program requirements
- Downloading CorpusSearch
- Key properties of CorpusSearch
- Some example searches
Program requirements

- A corpus without a search program is like the Internet without Google.
- Enter CorpusSearch (Randall 2000-2007), a dedicated search engine for parsed corpora
- Written in Java
- Runs under any Java-supported operating system (Linux, Mac, Unix, Windows)
- Requires Java 2, version 1.5 or later
- Expects labelled bracketing (Penn Treebank style)
- Files must not contain extraneous formatting characters
Downloading CorpusSearch

- Corpus Search is freely available from http://corpussearch.sourceforge.net/
- Don’t bother remembering the URL.
- Just google ”CorpusSearch” (one word), and go to the first hit.
- Downloading and installing the program takes 5 minutes.
Key properties of CorpusSearch

- Output is searchable
- Search functions are linguistically intuitive
- User can custom-define search expressions
- Searches can ignore irrelevant material (punctuation, interjections, parentheticals, traces, ...)


Further properties of CorpusSearch

- Lexicon feature (useful for lemmatization)
- “Search and replace” corpus revision
  - Speeds up quality control of extant corpora
  - Facilitates the construction of training corpora from POS-tagged corpora
- Generation of coding strings for multivariate analysis (Varbrul, SPSS, etc.)
- Graphical interfaces are available for basic searches of parsed and POS-tagged corpora
- Reconstructing discontinuous dependencies (under development)
The first dedicated query language for parsed corpora was \textit{tgrep} (Pito 1993).
\textit{tgrep} does not accept on its own output as input.
Increasingly refined research hypotheses thus generate complicated, error-prone monster queries.
Advanced treebank query languages like CorpusSearch or TigerSearch accept their own output as input, allowing complex queries to be broken up into sequences of simpler ones.
Such sequences fit naturally with the step-by-step process of empirical research.
Search functions are linguistically intuitive

- same instance
- $x$ exists
- $x$ (immediately) precedes $y$
- $x$ (immediately) dominates (only) $y$
- $x$ immediately dominates $y$ ignoring intervening $z$
- $x$ immediately dominates $y$ as $n$-th node
- $x$ has sister $y$
- $x$ has same index as $y$
Example parsed sentence

( (IP-MAT (PP (P A=))
   (NP (D =u)
      (N saillir)
      (PP (P de)
         (NP (DZ mon) (N enfance))))))
(PON ,)
(NP-SBJ (PRO je))
(EJ fus)
(VPP amene)
(PP (P a)
   (NP (NPR Lisle)))
(PONFP .)))
Two very simple queries

- A query without disjunction
  
  node: IP-MAT

  query: (NP-SBJ iDomsOnly PRO)

- A query with disjunction

  node: IP-MAT | IP-MAT-PRN

  query: (NP-SBJ iDomsOnly PRO)
Kleene star

node: IP-MAT*

query: (NP-SBJ iDomsOnly PRO)

▸ IP-MAT* matches IP-MAT | IP-MAT-PRN | IP-MAT-SPE | IP-MAT-PRN-SPE | ...
Asterisks in the input

( (IP-MAT (PP (P A=))
    (NP (D =u)
        (N saillir)
        (PP (P de)
            (NP (DZ mon) (N enfance))))))
(PON ,)
(NP-SBJ *pro*)
(EJ fus)
(VPP amene)
(PP (P a)
    (NP (NPR Lisle)))
(PONFP .)) (ID COMMYNES,4.26)
Matching asterisks in the input

- *pro* matches pro | proimp | respro | prozac | ...
- To match *pro*, we have to ”escape” each asterisk with a backslash (\).

node: IP-MAT*

query: (NP-SBJ iDomsOnly \*pro\*)
Quick quiz

- How do we search for sentences with overt expletive (PROIMP) and thematic (PRO) subjects?
- How about their silent counterparts?
  *pro*, *proimp*
- How about silent subjects more generally?
  *con*, *pro*, *proimp*
- How about silent subjects, but not traces?
  *T-1*, *ICH*-2
node: IP-MAT*

query: (NP-SBJ* iPrecedes VJ)
AND (VJ iPrecedes NP-OB1*)

▶ Yvain aime Gawains.
▶ Yvain croit que Lunete aime Gawains.
V2 - example and query

( (IP-MAT (NP-TMP (D Le) (NCS lendemain)))
  (VJ arriva)
  (NP-SBJ (D la) (ADJ bonne) (NCS royne)
   (NP-PRN (NPRS Marguerite)))
  (. .)))

node: IP-MAT*

query: (NP-TMP* iPrecedes VJ)
  AND (VJ iPrecedes NP-SBJ*)
V3 - example and query

( (IP-MAT (NP-TMP (D Le) (NCS lendemain)))
  (NP-SBJ (PRO elle))
  (VJ arriva)
  ( . . )))

node: IP-MAT*

query: (NP-TMP* iPrecedes NP-SBJ*)
  AND (NP-SBJ* iPrecedes VJ)
Adding structural information - Why

( (IP-MAT (CP-ADV ... 
   (IP-SUB ... 
      (NP-TMP (D le) (NCS lendemain)))
   (VJ arriva)
   (NP-SBJ (D la) (ADJ bonne) (NCS royne)
      (NP-PRN (NPRS Marguerite))
   (. .)))
)
Adding structural information - How

node: IP-MAT*

query: (IP-MAT* iDoms NP-TMP*)
  AND (IP-MAT* iDoms VJ)
  AND (IP-MAT* iDoms NP-SBJ*)
  AND (NP-TMP* iPrecedes VJ)
  AND (VJ iPrecedes NP-SBJ*)
Another way

query: (IP-MAT* iDoms NP-TMP*)
AND (NP-TMP* HasSister VJ)
AND (NP-TMP* HasSister NP-SBJ*)
AND (NP-TMP* iPrecedes VJ)
AND (VJ iPrecedes NP-SBJ*)
Making the queries more general

- Topic needn’t be a temporal NP, but could be some other category.
- Finite verb needn’t be a main verb, but could be a finite auxiliary or modal.
query: (IP-MAT* iDoms ADJP* | ADVP* | NP* | PP* | QP*)
AND (IP-MAT* iDoms AJ | EJ | MDJ | VJ)
AND (IP-MAT* iDoms NP-SBJ*)
AND (ADJP* | ADVP* | NP* | PP* | QP*
     iPrecedes AJ | EJ | MDJ | VJ)
AND (AJ | EJ | MDJ | VJ iPrecedes NP-SBJ*)
Definitions files

finite_verb: AJ | EJ | MDJ | VJ

subject: NP-SBJ*

topic: ADJP* | ADVP* | NP-OB1* | NP-OB2* | NP-MSR* | NP-TMP* | PP* | QP*

- Put these three lines in a file with a .def extension (say, gtrc.def)
- Definitions files facilitate consistency across queries
Query using definitions

node: IP-MAT*

def: gtrc.def

query: (IP-MAT* idoms topic)
   AND (IP-MAT* idoms finite_verb)
   AND (IP-MAT* idoms subject)
   AND (topic iPrecedes finite_verb)
   AND (finite_verb iPrecedes subject)
The ignore nodes feature - Why

Let’s say we want this sentence to count as V2.
The ignore nodes feature - How

node: IP-MAT*

def: gtrc.def

ignore_nodes: IP-MAT-PRN* | ITJ* | PON*

query: (IP-MAT* idoms topic)
    AND (IP-MAT* idoms finite_verb)
    AND (IP-MAT* idoms subject)
    AND (topic iPrecedes finite_verb)
    AND (finite_verb iPrecedes subject)

▶ The ignore nodes statement can be stored either in the query or in the definitions file.
Excluding clitics from consideration

( (IP-MAT (NP-RFL (PRO S’)))
 (PP (PRO en))
 (VJ alla)
 (NP-SBJ (D le) (NCS roy))
 (NP-TMP (D Le) (NCS lendemain))
 ( . . )))

The clitics in the example are part of the verbal complex - not topics.

We do not want the sentence to count as V2.
The negation operator

query: (IP-MAT* iDoms topic)
    AND (topic idoms !PRO)
    AND (IP-MAT* iDoms finite_verb)
    AND (IP-MAT* iDoms subject)
    AND (topic Precedes finite_verb)
    AND (finite_verb iPrecedes subject)

├─ Putting the clause that excludes clitics early speeds up the search.
├─ Excluding clitics requires changing ”iPrecedes” to ”Precedes” in the penultimate clause.
Excluding sentences with empty subjects

( (IP-MAT (NP-TMP (D Le) (NCS lendemain)))
  (NP-SBJ *pro*)
  (VJ arriva)
  ( . . )))
The revised query

query: (subject idoms !**)
    AND (IP-MAT* iDoms topic)
    AND (topic idoms !PRO)
    AND (IP-MAT* iDoms finite_verb)
    AND (IP-MAT* iDoms subject)
    AND (topic iPrecedes finite_verb)
    AND (finite_verb iPrecedes subject)

▶ Once again, the position of the first clause speeds up the search.
A tricky case

( (IP-MAT (NP-TMP (D Le) (NCS lendemain)))
    (PON ,)
    (IP-MAT-PRN (NP-OB1 (D ce) (NCS fait))
        (VJ sait)
        (NP-SBJ (Q tout) (D le) (NCS monde)))
    (PON ,)
    (VJ mourut)))
    (NP-SBJ (D la) (NCS royne))
    (. .)))

▶ We want both the IP-MAT and the IP-MAT-PRN to count as V2.
▶ But the very same query that makes the IP-MAT count as V2 ignores the IP-MAT-PRN.
▶ Solution: Two separate queries
▶ Caution: Don’t count the same examples twice.
(NP (NCS Merci)
  (PP (P de)
    (NP (DZ votre) (NCS attention)))
  (PONFP .)))