

Building and searching large parsed corpora of diachronic texts

Beatrice Santorini

University of Pennsylvania

<http://www.ling.upenn.edu/~beatrice/corpus-ling.html>

Deutsch Diachron Digital

10 December 2003

Overview of presentation

- Bad news, good news
- Building a parsed corpus
 - Goals and principles of syntactic annotation
 - Some examples
 - Implementation
- CorpusSearch, a dedicated search engine for parsed corpora
- Developments in the works

Appendices

- Appendix 1: Details of POS tagging
- Appendix 2: Details of syntactic annotation
- Appendix 3: From raw text to parsed token

The bad news

- Constructing a corpus is time-consuming
- Correct parse is often unclear
(even synchronically, let alone diachronically)
- Consistency is difficult to maintain
- Good help is hard to find

The good news

- Re: Time-consuming
 - Electronic corpora can be built in stages
 - Electronic corpora are searchable
 - With a parsed corpus, research hypotheses are more easily tested and refined
 - Results are more reliable and replicable
 - Different *kinds* of results become possible
- Re: Unclear parses
 - Parses need **not (!)** be correct to be useful

- Re: Consistency
 - Work can be shared by various research groups
 - Results become replicable across research groups
- Re: Labor shortage
 - State-of-the-art parsers will allow annotation to be divided into more and less highly skilled tasks
 - Advances in query language will help to automate corpus construction yet further

Goals and principles of annotation

- Corpus consists of straight-up ASCII
 - Syntactic annotation is represented as labeled bracketing
 - No internal formatting codes
 - No dependence on obsolescent software
- Annotated corpus = God's truth, **not**
 - The primary goal of our annotation is to facilitate searches for various constructions of interest.
 - The goal is **not (!)** to associate every sentence with a correct structural description.

Dealing with uncertainty and ambiguity

- As many syntactic categories as possible should have clear meanings so that the number of unclear cases is minimized.
- We try to avoid controversial decisions.
- To that end, we sometimes omit information.
 - VP boundaries
 - Subtle distinctions (adjectival vs. verbal passives, argument vs. adjunct PPs)
- In other cases, we use default rules.
 - Location of wh-traces
 - PP attachment (“when in doubt, attach high”)

An example of diachronic ambiguity

Still underlying OV phrase structure?

```
(PP (P until)
  (CP-ADV (C 0)
    (IP-SUB (NP-SBJ (N death))
      (DOP do)
      (VP (NP-OB1 (PRO us))
        (VB part))))))
```

Or already underlying VO?

If so, does the pronoun move out of VP, as illustrated here, or remain within VP?

```
(PP (P until)
  (CP-ADV (C 0)
    (IP-SUB (NP-SBJ (N death))
      (DOP do)
      (NP-1 (PRO us))
      (VP (VB part)
        (NP-OB1 *T*-1))))))
```

Omitting undecidable information

Our solution: a 'flat' structure **without** a VP

```
(PP (P until)
  (CP-ADV (C 0)
    (IP-SUB (NP-SBJ (N death))
      (DOP do)
      (NP-OB1 (PRO us))
      (VB part))))
```

Another example

- ((CP-QUE (WNP-1 Which house)
 (IP-SUB (MD did)
 (NP-SBJ they)
 (NP-OB1 *T*-1)
 (VB buy)
 (. ?)))
- ((CP-QUE (WNP-1 Which house)
 (IP-SUB (MD did)
 (NP-SBJ they)
 (VB buy)
 (NP-OB1 *T*-1)
 (. ?)))

An incorrect, yet useful, structure

Our solution: we consistently put the trace in a position that is **linguistically unmotivated**.

```
( (CP-QUE (WNP-1 Which house)
      (IP-SUB (NP-OB1 *T*-1)
              (MD did)
              (NP-SBJ they)
              (VB buy)
              (. ?))))
```

An example from the EModEng corpus

Points of interest (see next slide)

- Expletive *there* is coindexed with logical subject
- Annotation indicates where (silent) relative pronoun is interpreted
- Tokens are identified by reference labels

```
ALHATTON  2,      241.  7
```

```
text ID    vol.   page  serial token number
```

Volume number is optional; serial token number is unique within text.

Example sentence 1

```

( (IP-MAT (NP-SBJ=1 (EX There))
  (BEP is)
  (NP-1 (ONE one) (NPR M=r=) (NPR Colson)
    (CP-REL (WNP-2 0)
      (C 0)
      (IP-SUB (NP-SBJ (PRO I))
        (BEP am)
        (ADJP (ADJ shure)
          (CP-THT (C 0)
            (IP-SUB (NP-ACC *T*-2)
              (NP-SBJ (PRO$ my) (N Lady))
              (HVP has)
              (VBN seen)
              (PP (P at)
                (NP (N diner)
                  (PP (P w=th=)
                    (NP (PRO$ my)
                      (N Unckle))))))
            (. .)) (ID ALHATTON,2,241.7))

```

A second example from the EModEng corpus

Points of interest (see next slide)

- Annotation indicates dependency between measure phrase (*so much*) and degree complement clause
- Locative (as well as directional and temporal) AdvPs are specially marked.

Example sentence 2

```
( (IP-MAT (NP-SBJ (PRO I))
  (HVP have)
  (NP-ACC (NP-MSR (QP (ADVR so) (Q much)
    (CP-DEG *ICH*-1))))
  (N buisness)
  (ADVP-LOC (ADV here))
  (CP-DEG-1 (C y=t=)
    (IP-SUB (NP-SBJ (PRO I))
      (VBP hope)
      (CP-THT (C 0)
        (IP-SUB (NP-SBJ (PRO$ my) (N Lady))
          (MD will)
          (VB excuse)
          (NP-ACC (PRO me))
          (PP (P till)
            (NP (ADJS next)
              (N post))))))))))
(. .)) (ID ALHATTON,2,245.46))
```

<----- not a typo!

How we build a parsed corpus - a flowchart

- POS tagging
 - Automatic preprocessing (punctuation, contractions)
 - Automatic tagging (Brill 1995)
 - Human correction
- Parsing
 - Automatic parsing (Collins 1996)
 - Human editing (= correction + addition of information)
- Final editing (partially automated)

Correction software

- We use correction software developed in connection with the Penn Treebank (<http://www.cis.upenn.edu/~treebank>) and implemented in Emacs Lisp
- Incorrect tags are corrected by positioning cursor on item to be corrected and entering correct tag
- Proposed tag is checked to ensure that new tag is legal
- Correction software leaves input text inviolate

Project management

- Mean editing speed (in language well-known to annotator):
2,000 words/hours for POS-tagging
1,000 words/hours for parsing
- Annotators can work approx. 4 hours/day or 20 hours/week
- Annotators are relatively easy to find and train for POS-tagging, but quite a bit harder to find and train for parsing (people are used to thinking about words, but not in terms of constituent structure)

So how long does it take to produce a parsed corpus of 1 M words?

- POS-tagging stage
 - 1,000,000 words / 2,000 words/hours = 500 hours
 - 500 hours / 20 hours/week = 25 weeks
- Parsing stage
 - 1,000,000 words / 1,000 words/hours = 1,000 hours
 - 1,000 hours / 20 hours/week = 50 weeks
- Total: 75 weeks

A search engine for parsed corpora

- A corpus without a search program is like the Internet without Google.
- Enter CorpusSearch (Randall 2000), a dedicated search engine for parsed corpora
- Written in Java
- Runs under Linux, Mac, Windows, Unix

Properties of CorpusSearch

- A key feature: The output of CorpusSearch is searchable
- Basic search functions are linguistically intuitive
- End user can custom-define further linguistically relevant search expressions
- Searches can disregard material (interjections, parentheticals, traces, ...)
- A cool feature: CorpusSearch can produce coding strings

A key feature: Searchable output

- Complicated and error-prone monster queries can be implemented as a sequence of simpler queries.
- Sequences of queries are consistent with the way that corpus research proceeds, via a successive refinement of hypotheses.
- Generating searchable output slows CorpusSearch down somewhat (searches of 1-2M words can take 2-3 minutes)

Basic search functions are linguistically intuitive

- exists
- precedes
- immediately precedes
- immediately dominates

Simple **dominate** discontinued in CorpusSearch 1.1, but easy to simulate

A simple sample query

node: IP*

query: (IP* iDoms NEG)

- Asterisk is a wildcard
(IP* matches IP-MAT, IP-SUB, IP-INF, etc.)
- CorpusSearch searches the corpus for constituents with the label(s) specified in **node**.
- Whenever it finds such a constituent, it checks whether the material in the constituent matches the condition(s) in **query**.
- Matching tokens are recorded in an output file.

Complement output files

- Sometimes we are interested in tokens that *don't* match a query. If desired, CorpusSearch records such tokens in a **complement** file.

```
print_complement: true
```

```
node: IP*
```

```
query: (IP* iDoms NEG)
```

Fine-tuning searches

((CP-QUE Did they see any snow leopards?))

((IP-MAT (NP Not a single one) did they see.))

((NP Not a single one).

- Query 1 returns only the second token:

node: IP*

query: (NP* iDoms NEG)

- Query 2 returns both the second and the third tokens:

node: NP*

query: (NP* iDoms NEG)

Definition files

- Let's say we want to find NP objects in the Vorfeld:
 - Den Lothar habe ich recht gern.
 - Dem Lothar ist nicht zu trauen.
 - Des Lothars gedenken wir selten.
- We don't want other constituents in the Vorfeld:
 - Nächste Woche treffe ich den Lothar.
 - Mit dem Lothar verstehe ich mich gut.
 - Nur selten gedenken wir Lothars.

A possible query, but long-winded and error-prone

node: S*

```
query: ((S* iDomsNum1 NP-AKK | NP-DAT | NP-GEN)
        AND
        (S* iDomsNum2 SEIN-PRS | SEIN-PRT |
          HAB-PRS | HAB-PRT |
          MODAL-PRS | MODAL-PRT |
          VERB-PRS | VERB-PRT))
```

A better way

define: v2.def

node: S*

query: ((S* iDomsNum1 Objekt)
AND
(S* iDomsNum2 Vb-fin))

Contents of the definition file v2.def:

Objekt: NP-AKK | NP-DAT | NP-GEN

Vb-fin: SEIN-PRS | HAB-PRS | MODAL-PRS | VERB-PRS |
SEIN-PRT | HAB-PRT | MODAL-PRT | VERB-PRT

Using definition files to construct word classes

- Definition file defines:

```
verb:      VB | VBP | VBD | VAG | VAN
```

```
drink:     drink | drinks | drank | drunk | drinking
```

```
eat:       eat | eats | ate | eaten | eating
```

```
read:      read | reads | reading
```

```
write:     write | writes | wrote | written | writing
```

```
TR-INTR:  \ $drink | \ $eat | \ $read | \ $write
```

- Query file makes reference to word class:

```
query: (verb iDoms TR-INTR)
```


Ignoring material

- CorpusSearch ignores certain material by default.
 - punctuation
 - page numbers
 - editorial comments
- The default is overridable.
- In addition, other material can be ignored as convenient or necessary.

Ignoring material in searching for V3

- We certainly want this sentence to count as V3:
 - (PP In Jahre 1356,) (NP-SBJ der Kaiser) besuchte den Papst.
- But we might want these sentences to count as V2:
 - (CONJ Und) (NP-AKK den Lothar) treffe ich morgen.
 - (INTJP Ojemine,) (NP-AKK den Lothar) treffe ich wohl nicht mehr.
 - (NP-LFD-1 Den Lothar,) (NP-AKK=1 den) treffe ich morgen.
 - (NP-VOC Du,) (NP-AKK den Lothar) treffe ich erst im Januar.

The V3 query

definition: v2.def

node: S*

add_to_ignore: CONJ | INTJP | NP-LFD* | NP-VOC

query: (S* iDomsNum3 Vb-fin)

The asterisk after NP-LFD is necessary because left-dislocated constituents are coindexed with resumptive expressions,

A cool feature: The coding function

- Ordinary queries generate one output file per query.
- This can lead to an unwieldy proliferation of output files.
- Moreover, we are often interested in multivariate statistical analysis.

An example from this history of English

- Verb raising (old grammar):
He loves me, he loves me not.
- *Do* support (new grammar):
She loves me, she does not love me.

Factors for multivariate analysis (for illustration only)

- Date of text: period 1, 2, 3
- Author's sex: f)emale, m)ale
- Text genre: b)ible, n)onvernacular, v)ernacular

A coding query

```
1: { o: (finite_verb precedes NEG)
     n: ((finite_do precedes NEG) AND (NEG precedes VB)) }

2: { 1: (*AMBASS* inID)
     3: (*ANHATTON* inID)
     1: (*APOOLE* inID)
     2: (*ARMIN* inID)
     3: (*AUNGIER* inID)
     2: (*AUTHNEW* inID)
     2: (*AUTHOLD* inID) }

3: { f: (*ANHATTON* inID)
     f: (*APOOLE* inID)
     m: ELSE }

4: { v: (*ANHATTON* inID)
     v: (*APOOLE* inID)
     v: (*ARMIN* inID)
     b: (*AUTHNEW* inID)
     b: (*AUTHOLD* inID)
     n: ELSE }
```

- See handout for output of coding query.
- Coding strings can include blank slots for subsequent manual analysis. This is useful in analyzing discourse factors.
- Coding strings are compatible with standard programs for multivariate analysis of linguistic variation.

- Output of coding queries can serve as input to subsequent coding queries. This is useful for analyzing statistical interaction.

For purposes of illustration, the following query groups nonvernacular texts written by women as distinct from all other combinations of genre and sex.

```
5: {  
    a: ((CODING col4 n) AND (CODING col3 f))  
    b: ELSE  
}
```

Developments in the works - CorpusSearch

- Better negation
- Better disjunction
- Extended support for regular expressions
- “Search and replace” annotation support

Negation

- Negation in CorpusSearch 1.1 cannot take scope over search functions. To find sentences without an object, you **cannot** use the following query.

```
query: (S* !iDoms Objekt)
```

- The next query is legal, but doesn't necessarily do what users expect.

```
query: (S* iDoms !Objekt)
```

This query returns all sentences that contain **something other than** an object, not sentences that don't contain an object.

- To find sentences without an object, you have to search for sentences **with** an object and produce the complement file.

```
print_complement: true  
query: (S* iDoms Objekt)
```

- CorpusSearch 2 will allow negation over predicates, giving the desired output directly.

Disjunction

- In CorpusSearch 1.1, conditions in ordinary queries cannot be connected by OR.

```
query: ((condition_1) OR (condition_2))
```

- Disjunctive queries **can** be simulated by using coding queries.

```
1: {  
    x: (condition_1)  
    x: (condition_2)  
}
```

- CorpusSearch 2 will support OR (both inclusive and exclusive) in ordinary queries.

Extended support for regular expressions

- CorpusSearch 1.1 requires:

```
kein:      kein | Kein | keyn | Keyn
keine:     keine | Keine | keyne | Keyne
keinem:    keinem | Keinem | keynem | Keynem
keinen:    keinen | Keinen | keynen | Keynen
keiner:    keiner | Keiner | keyner | Keyner
keines:    keines | Keines | keynes | Keynes
KEIN:      $kein | $keine | $keinem | $keinen |
           $keiner | $keines
```

- CorpusSearch 2 will allow:

```
kein:      [kK]e[iy]ine*[mnrs]*
```

Search and replace annotation support

- Our annotation scheme requires subordinate clauses to contain a complementizer (= subordinating conjunction), possibly silent.
- We do this to give the following pairs of sentences parallel structures:
 - I know _ you are coming.
I know that you are coming.
 - They wonder when _ you are coming.
They wonder when that you are coming. (cf. Bavarian)

- Currently, silent complementizers need to be added by hand or with Perl scripts.
- In order to automate annotation yet further, CorpusSearch 2 will include [search and replace](#) annotation support.

Developments in the works - Parsing

- A parser being developed at Penn (Bikel, dissertation in progress) lets the user modify linguistic parameters, allowing increased crosslinguistic flexibility.
- The same parser allows multiple passes through a corpus, each pass respecting the previous ones.
- Advantages of multiple pass syntactic annotation
 - Multiple passes simplify the editing task (“divide et impera”)
 - Simplification means improvements in speed and consistency
 - Editing could be carried out by a mixture of more and less highly trained annotators.

Appendix 1: Details of POS tagging stage

POS tagging - Automatic stage

- Step 1: Text is tokenized
 - Punctuation is split off from words
 - Contractions are decomposed into their constituents
we'll → \$we/PRO \$'ll/MD {TEXT:we'll}/CODE
- Step 2: Text is run through tagger (in our case, Brill 1995)

The Brill tagger

- Step 1:

Based on a training corpus (= a relatively large corpus of already tagged text), each word is tagged with its most frequent part of speech

He/PRO opened/VBD a/D can/MD of/P soup/N

- Step 2:

Tagger guesses at the tag for words that are not in the training corpus

Wimple/? → Wimple/NPR

wimple/? → wimple/N

The Brill tagger, 2

- Step 3:

Tagger refines guesses from Step 2 on the basis of morphological clues

wimpleless/**N** → wimpleless/**ADJ**

- Step 4:

Tagger adjusts tags from Step 1 in light of context

. . . a/D can/**MD** of/P soup/N → . . . can/**N** . . .

POS tags

- List of POS tags
<http://www.ling.upenn.edu/~ataylor/ppcme-lite.htm#labellists>
- Tagset relatively small because
 - Large tagsets lead to sparse data problem
 - Our focus is on parsed, rather than tagged, data

Training the POS tagger

- The Brill tagger assigns tags based on the statistical analysis of a training set.
- It is possible to retrain the tagger. For instance, we began tagging the EModEng corpus with the tagger trained on late MidEng data. Once we had corrected 500K words of EModEng, we used that data as a training set for the rest of the corpus.
- Training the Brill tagger on a large training set is time-consuming (for 500K words, approx. 11 days of computer time)

Appendix 2: Details of parsing stage

Parsing - Automatic stage

- POS-tagged text is stripped of all but correct tags
- Text is run through a parser (in our case, Collins 1996)
- Output of parser is in the form of formatted labeled bracketing, in which depth of indenting corresponds to depth of embedding

The Collins parser

- Parses strings according to structures mostly frequently associated to input in a training corpus
- Has access to both POS tags and to lexical items to choose likely attachment
 - draw the man with the pen (high attachment)
 - draw the man with the sandwich (low attachment)
- Like the Brill tagger, the Collins parser can be trained.

Syntactic annotation - some useful URLs

- List of syntactic tags

<http://www.ling.upenn.edu/~ataylor/ppcme-lite.htm#labellists>

- Annotation manual

- For beginners

<http://www.ling.upenn.edu/~ataylor/ppcme-lite.htm>

- For advanced users

<http://www.ling.upenn.edu/~ataylor/ppcme2-man-toc.htm>

Parsing - Human editing stage

Editing operations include:

- Changing syntactic tags
- Breaking up run-on sentences or consolidating fragments
- Adding subcategory information
 - ADVP → ADVP-TMP
 - CP → CP-CMP
- Changing attachment level

Parsing - Human editing stage, 2

- Adding empty categories
- Adding indices to
 - an antecedent and its trace
 - expletive expressions ('it', 'there') and their associates
 - a constituent in a noncanonical position and the position in which it is interpreted

Appendix 3: From raw text to parsed token

Raw text (reflecting Helsinki Corpus markup)

Helsinki uses # to indicate continuation of line in source edition.

My Lord,

I return my most humble thankes for y=e= honour of y=r= #
Lord=ps= letter.

I have not yet bin any were, but at shopes and a veseting; but #
I

believe shall be on Munday at a ball at St. Jeames, where, as #
they

tell me, ther is a famose new danser to apere, which is to #
charme

us all, but not make amends for y=e= loss of M=rs= Ibbings who #
danced

at Lincolns Inn Feild and is lately dead.

Tokenized text

My Lord , I return my most humble thankes for y=e= honour
of y=r= Lord=ps= letter .

I have not yet bin any were , but at shopes and a veseting ;
but I believe shall be on Munday at a ball at St. Jeames ,
where , as they tell me , ther is a famose new danser to
apere , which is to charme us all , but not make amends
for y=e= loss of M=rs= Ibbings who danced at Lincolns
Inn Feild and is lately dead .

Tagged text before correction

Tagger errors are highlighted in red. The narrow text formatting facilitates human correction.

My/PRO\$ Lord/N ,/, I/PRO
return/VBP my/PRO\$ most/QS
humble/ADJ thanks/NS for/P
y=e=/D honour/N of/P
y=r=/PRO\$ Lord=ps=/N\$
letter/N ./.

I/PRO have/HVP not/NEG
yet/ADV bin/BEN any/Q
were/**BED** ,/, but/P at/P
shope/NS and/CONJ a/**D**
veseting/**VAG** ;/.

but/CONJ I/PRO believe/VBP
shall/MD be/BE on/P
Munday/NPR at/P a/D ball/N
at/P St./NPR Jeames/NPR ,/,

where/WADV ,/, as/P they/PRO
tell/VBP me/PRO ,/, ther/EX
is/BEP a/D famose/ADJ new/ADJ
danser/N to/TO apere/VB ,/,
which/WPRO is/BEP to/TO
charme/VB us/PRO all/Q ,/,
but/**P** not/NEG make/VB
amends/NS for/P y=e=/D loss/N
of/P M=rs=/NPR Ibbings/NPR
who/WPRO danced/VBD at/P
Lincolns/**NPR** Inn/NPR
Feild/NPR and/CONJ is/BEP
lately/ADV dead/ADJ ./.

Tagged text after correction

Tagger errors are highlighted in red; human corrections in green.

My/PRO\$ Lord/N ,/, I/PRO return/VBP my/PRO\$ most/QS humble/ADJ
thankes/NS for/P y=e=/D honour/N of/P y=r=/PRO\$ Lord=ps=/N\$
letter/N ./ . I/PRO have/HVP not/NEG yet/ADV bin/BEN any/Q
were/**BED***/**WADV** ,/, but/P at/P shope/NS and/CONJ a/**D***/**P**
veseting/**VAG***/**N** ;/. but/CONJ I/PRO believe/VBP shall/MD be/BE on/P
Munday/NPR at/P a/D ball/N at/P St./NPR Jeames/NPR ,/, where/WADV
/, as/P they/PRO tell/VBP me/PRO ,/, ther/EX is/BEP a/D famose/ADJ
new/ADJ danser/N to/TO apere/VB ,/, which/WPRO is/BEP to/TO
charme/VB us/PRO all/Q ,/, but/**P***/**CONJ** not/NEG make/VB amends/NS
for/P y=e=/D loss/N of/P M=rs=/NPR Ibbings/NPR who/WPRO
danced/VBD at/P Lincolns/**NPR***/**NPR\$** Inn/NPR Feild/NPR and/CONJ
is/BEP lately/ADV dead/ADJ ./ .

Parsed text, before correction

```
( (IP-MAT (NP-SBJ (PRO$ My) (N Lord)) <--- should be NP-VOC
  (, ,)
  (NP-SBJ (PRO I))
  (VBP return)
  (NP-ACC (PRO$ my) (QS most) (ADJ humble) (NS thanks)) <----
  (PP (P for)                                we bracket multiword AdjPs
    (NP (D y=e=) (N honour)
      (PP (P of)
        (NP (NP-GEN (PRO$ y=r=) (N$ Lord=ps=))
          (N letter))))))
  (. .)))
```

```

( (IP-MAT (NP-SBJ (PRO I))
  (HVP have)
  (NEG not)
  (ADVP (ADV yet)) <---- we label temporal AdvPs
  (BEN bin)
  (NP-ACC (Q any))
  (CP (WADVP (WADV were)) <---- parser misled by unusual word boundary
    (, ,)
    (C 0)
    (PP (P but) (P at) <---- parser treats 'but at' like 'out of'
      (NP (NS shopen)
        (CONJP (CONJ and)
          (PP (P a)
            (NP (N veseting))))))))
  (. ;)))

```

Parsed text, after correction

```
( (IP-MAT (NP-VOC (PRO$ My) (N Lord))
  (, ,)
  (NP-SBJ (PRO I))
  (VBP return)
  (NP-ACC (PRO$ my) (ADJP (QS most) (ADJ humble)) (NS thanks)
    (PP (P for)
      (NP (D y=e=) (N honour)
        (PP (P of)
          (NP (NP-GEN (PRO$ y=r=) (N$ Lord=ps=))
            (N letter)))))))
  (. .)) (ID ALHATTON,2,240.5))
```

```

( (IP-MAT (NP-SBJ (PRO I))
  (HVP have)
  (NEG not)
  (ADVP-TMP (ADV yet))
  (BEN bin)
  (ADVP-LOC (Q any) (WADV were)
    (, ,)
    (PP (P but)
      (PP (PP (P at)
        (NP (NS shopen))
        (CONJP (CONJ and)
          (PP (P a)
            (NP (N veseting))))))))))
  (. ;)) (ID ALHATTON,2,240.6))

```