SYNTACTIC ANNOTATION IN HISTORICAL CORPUS DATA

SUM-UP Summer School 2016

July 7, 2016

Hagen Hirschmann
Humboldt-Universität zu Berlin
hirschhx@hu-berlin.de

Material

- Please download material at hu.berlin/WSdata

- Please access ANNIS search interface with your web browser: korpling.german.hu-berlin.de/annis3/

Corpus linguistics

- Corpus linguistics → empirical method for linguistic descriptions, theoretic modelling, …
- Possible aims
  - Finding exemplars for specific structure
  - Finding evidence for language use
  - Identifying typical structures in specific contexts
  - Quantifying competing structures
  - Tracking developments of grammatical structures
- Definition of "corpus": Digitally available and searchable textual data
  - often: annotations
  - often: metadata

Corpus linguistics - fields

- Collection of corpus data
- Preparation and processing of corpus data
- Analyzes of corpus data

Corpus linguistics - fields

- Collection of corpus data
  - Crawling internet
  - Collecting analog texts
  - Storing raw data in consistent format
- Preparation and processing of corpus data
- Analyzes of corpus data

Corpus linguistics - fields

- Collection of corpus data
  - Crawling internet
  - Collecting analog texts
  - Storing raw data in consistent format
- Preparation and processing of corpus data
  - **Digitizing data or transcribing raw data**
  - **Normalizing data**
  - **Annotating data**
  - **Storing annotated data in searchable format (search engine environment)**
- Analyzes of corpus data

Corpus linguistics - fields

- Collection of corpus data
  - Crawling internet
  - Collecting analog texts
  - Storing raw data in consistent format
- Preparation and processing of corpus data
  - **Digitizing data or transcribing raw data**
  - **Normalizing data**
  - **Annotating data**
  - **Storing annotated data in searchable format (search engine environment)**
- Analyzes of corpus data
  - **Formulating corpus queries**
  - **Further processing of relevant corpus data: statistical analyses**

Corpus linguistics - fields

- Collection of corpus data
  - Crawling internet
  - Collecting analog texts
  - Storing raw data in consistent format
- Preparation and processing of corpus data
  - **Digitizing data or transcribing raw data**
  - **Normalizing data**
  - **Annotating data**
  - **Storing annotated data in searchable format (search engine environment)**

Aim: syntactic annotations of corpus text

- Analyzes of corpus data
  - **Formulating corpus queries**
  - **Further processing of relevant corpus data: statistical analyses**

Corpus linguistics - fields

- Collection of corpus data
  - Crawling internet
  - Collecting analog texts
  - Storing raw data in consistent format
- Preparation and processing of corpus data
  - **Digitizing data or transcribing raw data**
  - **Normalizing data**
  - **Annotating data**
  - **Storing annotated data in searchable format (search engine environment)**

1st aim: syntactic annotations of corpus text

- Analyzes of corpus data
  - **Formulating corpus queries**
  - **Further processing of relevant corpus data: statistical analyses**

2nd aim: systematic searches in syntactically annotated corpus data

Syntactically annotated historical corpora (German)

- Reference corpus for Old High German (OHG) (Jena, Frankfurt, Berlin) (DDD project) http://www.deutschdiachrondigital.de/
  - Aim: annotate ~650000 word tokens from ~750-1050 on various grammatical levels
  - Archived and searchable via ANNIS (see below)
  - Currently ~350000 word tokens processed and available in ANNIS (https://korpling.german.hu-berlin.de/annis3/ddd)
  - Syntactic annotations in span format:
    - POS
    - Sentence types (such as subject clause, adverbial clause, …; no constituents etc.)
  - Schema developed as a standard for all language stages

Syntactically annotated historical corpora (German)

- Reference corpus for Middle High German (MHG) (Bochum, Bonn) http://referenzkorpus-mhd.uni-bonn.de/
  - Aim: annotate ~2100000 annotated word tokens from ~1050-1350 on various grammatical levels
  - Archived and searchable via ANNIS (see below)
  - Currently ~350000 word tokens processed and available in ANNIS (https://korpling.german.hu-berlin.de/annis3/ddd)
  - Syntactic annotations in span format:
    - POS
    - Sentence types (such as subject clause, adverbial clause, …; no constituents etc.)

Syntactically annotated historical corpora (German)

- Reference corpus for Early New High German (ENHG) (Bochum, Halle, Potsdam) http://www.ruhr-uni-bochum.de/wegera/ref/index.htm
  - Aim: annotate 4400000 annotated word tokens from ~1350-1650 on various grammatical levels
  - Archived and searchable via ANNIS (see below)
  - Currently ~350000 word tokens processed and available in ANNIS (https://korpling.german.hu-berlin.de/annis3/ddd)
  - Syntactic annotations in span format:
    - POS
    - Sentence types (such as subject clause, adverbial clause, …; no constituents etc.)

Syntactically annotated historical corpora (German)

- Ridges project (**R**egister **i**n **D**iachronic **Ge**rman **S**cience): Annotation of ENHG herb texts (HU Berlin) http://korpling.german.hu-berlin.de/ridges/index_en.html
  - Currently ~185000 annotated tokens processed and available in ANNIS (https://korpling.german.hu-berlin.de/annis3/ddd)
  - Extensive pipeline of **normalization** (see later)
  - Syntactic annotations in span format:
    - POS
    - Verb position
    - Subordinate clause type
  - Future: Full sentence parses (LangBank project; http://sfs.uni-tuebingen.de/langbank/)

Syntactically annotated historical corpora (German)

- DDB (deutsche diachrone Baumbank, German diachronic treebank; finished project) http://korpling.german.hu-berlin.de/ddb-doku/
  - Old High German (OHG), Middle High German (MHG), Early New High German (ENHG)
  - Aim: homogeneous syntactic annotations for different language stages
  - Archived and searchable via ANNIS (see below)
  - ~8300 tokens in all language stages, ~2800 per language stage
  - Syntactic annotations:
  - POS
  - Sentence constituent structure and functional categories in tree format: TIGER trees

Example annotation: DDD-Monsee

⊟ annotations

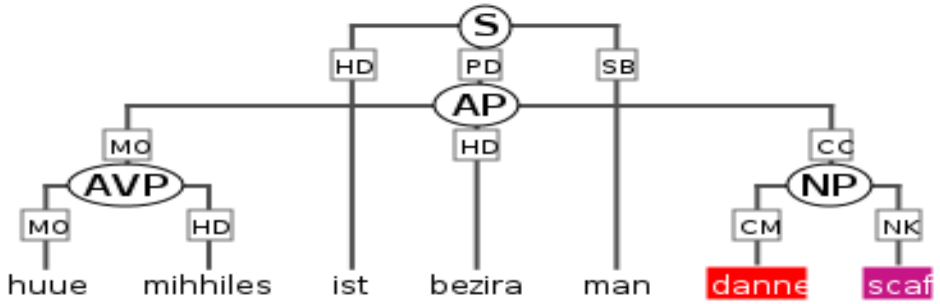
edition	huuemihhiles	ist	bezira	man	·	danne	scâf	·
ling	huuemihhiles	ist	bezira	man	·	danne	scâf	·
lemma	wio mihhil	sīn	beʒʒiro	man		danne	scāf	
posLemma	ADV	VA	ADJ	NA	$.	KO	NA	$.
pos	ADV	VAFIN	ADJD	NA	$.	KOKOM	NA	$.
inflectionClassLemma		IRR	N	C,A_MASC			A_NEUT	
inflectionClass		IRR	N	C,A_MASC			A_NEUT	
inflection		IND_PRES_SG_3	COMP_NEUT_SG_NOM_WK	SG_NOM			SG_NOM	
lang	goh	goh	goh	goh		goh	goh	
clause	CF_U_Int							
document	MF_Monsee-fragment-1_M.IV							
page	5	7						
line	25	26						
chapter	IV							
translation	um wieviel mehr	sein	besser	Mann, Mensch		denn, als	Schaf	

CF → finite clause
U → not complementized
Int → interrogative

Example tree: DDB-Monsee

huue	mihhiles	ist	bezira	man	danne	scaf
wio	mihhil	wesan	guot	man	thanne	scaf
--	--	3.Sg.Pres.Ind	Comp.Nom.Sg.Neut	Nom.Sg.Neut	--	Nom.Sg.Neut
PWAV	ADV	VAFIN	ADJD	NN	KOKOM	NN

⊟ tree



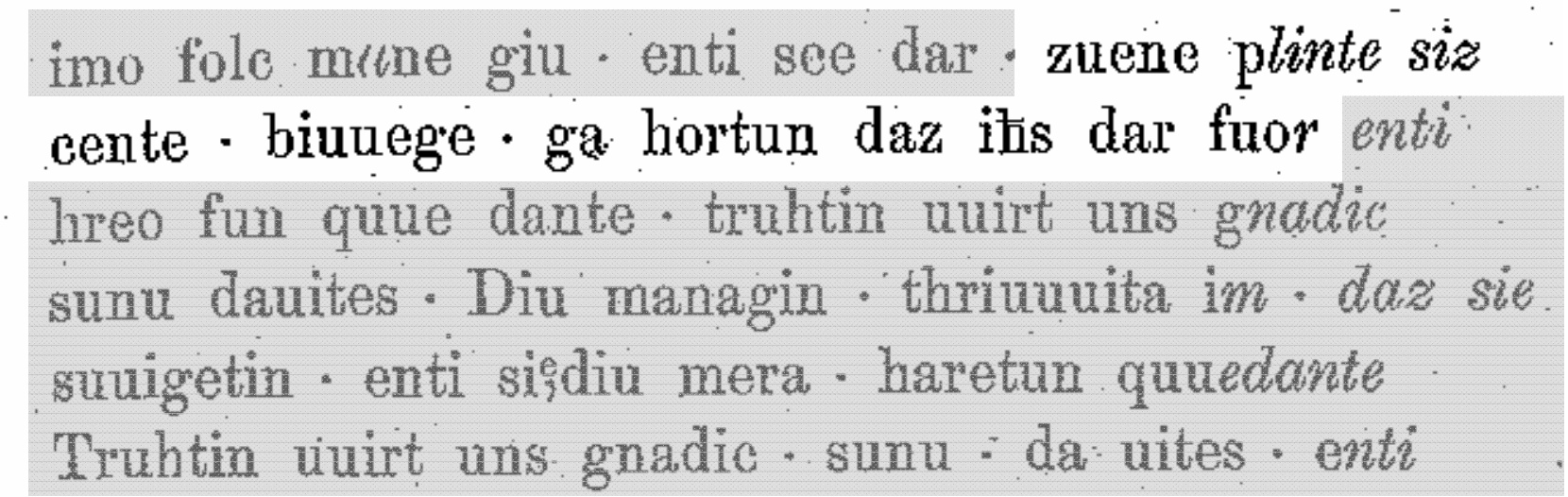
⊟ exmaralda

bib	Hench/5/M-IV/25		Hench/7/M-IV/26				
edition	huuemihhiles		ist	bezira	man|▪	danne	scaf|▪
lang	Ahd						
latin	(12) quanto magis / melior est homo oue?						
tok	huue	mihhiles	ist	bezira	man	danne	scaf

Questions for today

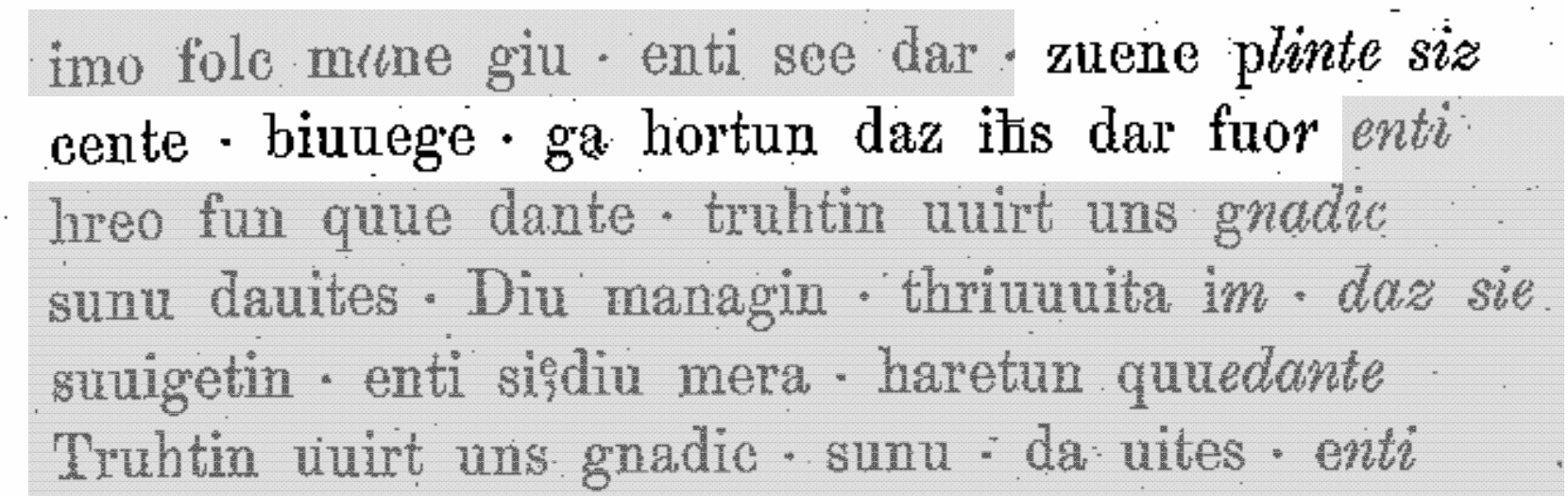
- How can we analyze historical texts syntactically in a holistic way?
- How do have to prepare the textual data for a holistic syntactic analysis?

Syntax analysis on basis of original text edition (Hench 1890)

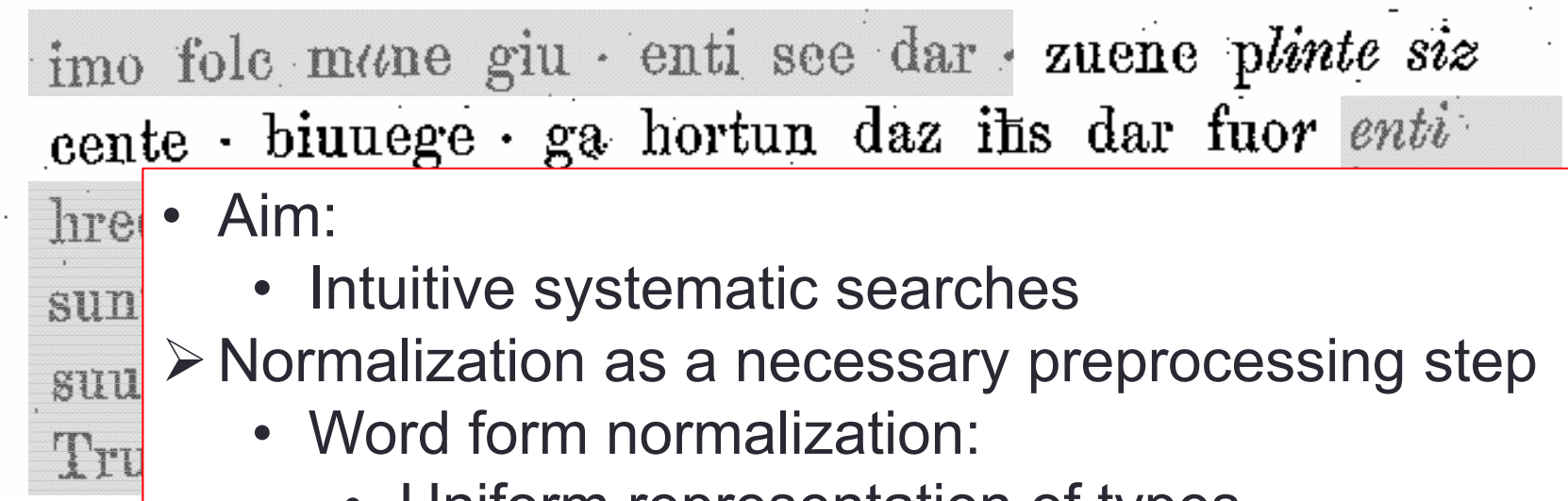

imo folc mane giu · enti see dar · zuene plinte siz
cente · biuuege · ga hortun daz ihs dar fuor enti
hreo fun quue dante · truhtin uuirt uns gnadic
sunu dauites · Diu managin · thriuuuita im · daz sie
suuigetin · enti sidiu mera · haretun quuedante
Truhtin uuirt uns gnadic · sunu · da uites · enti

- Phrasal status of *biuuege*?
- Syntactic status of *ga hortun*?
- Finding all instances of *Jesus*?

Aspects of normalization


imo folc mænne giu · enti see dar · zuene plinte siz
cente · biuuege · ga hortun daz ihs dar fuor enti
hreo fun quue dante · truhtin uuirt uns gnadic
sunu dauites · Diu managin · thriuuuita im · daz sie
suuigetin · enti sigdiu mera · haretun quuedante
Truhtin uuirt uns gnadic · sunu · da uites · enti

- Text format
- Word separation
- Writing inconsistencies
- Ligatures

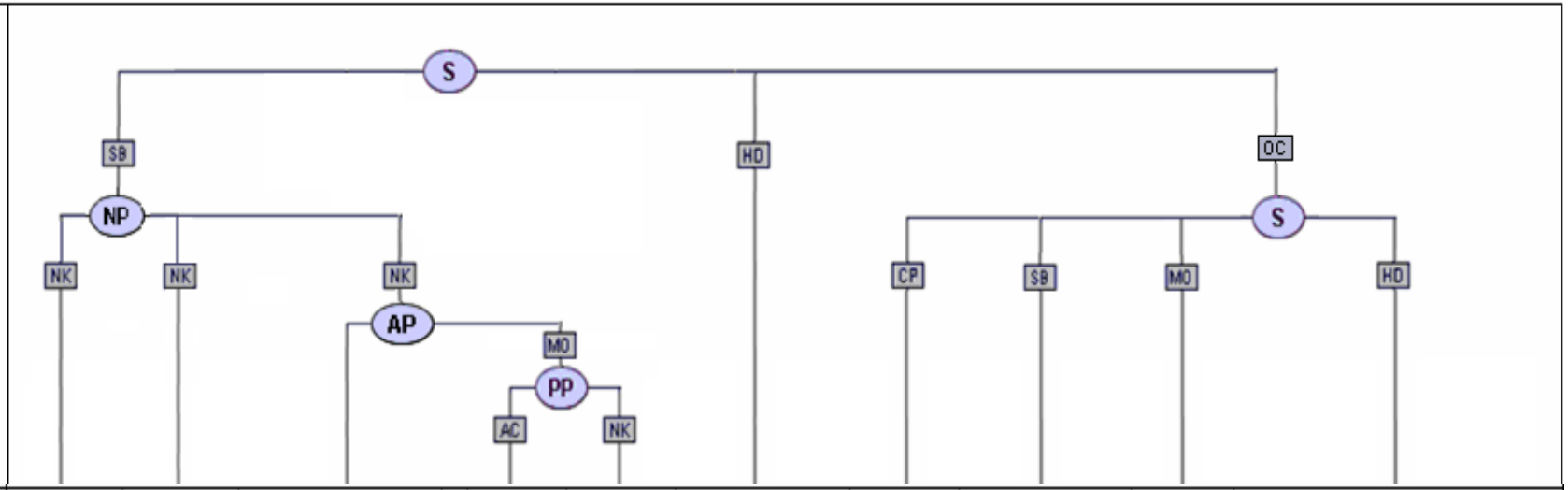


- Aim:
  - Intuitive systematic searches

➢ Normalization as a necessary preprocessing step

- Word form normalization:
  - Uniform representation of types
  - Word separation normalization
  - Normalization of punctuation
  - Hyper lemmata in diachronic data
- Important decision: Which reference of normalization?

Example of complex annotations DDB: Monsee fragments

Syntax											
Normal.	zuene	plinte	sizsente		bi	uuege	gahortun	daz	iesus	dar	fuor
Hench Edit.	zuene	*plinte*	*siz* / sente	•	biuuege		ga hortun	daz	ihs	dar	fuor
Lemma	zwene	blint	sizzen		bi	weg	gihoren	thaz	iesus	thar	faran
POS	CARD	NN	ADJD		APPR	NN	VVFIN	KOUS	NE	ADV	VVFIN
Morph.	Nom.Pl. Masc	Nom.Pl. Masc	Nom.Pl. Masc			Dat.Sg. Masc	3.Pl.Past.Ind		Nom.Sg.Masc		3.Sg.Past.Ind
Latein	duo caeci sedentes secus uiam audierunt, quia iesus transiret										

Steps of preprocessing: Making historical data (manually or automatically) parsable

- (parsable = syntactically interpretable)
- Tokenization: separating text into elementary units
  - For syntactic annotation, tokens should be syntactic words
- Normalization:
  - Treating reoccurring units equally (spelling variation; compare Logačev, Goldschmidt, Demske 2014),
  - Erasing structures that hinder syntactic interpretation
  - "Translating" elements into modern language for which we have adequate methods for a syntactic description

Steps of preprocessing: Making historical data (manually or automatically) parsable

- But: Never erase unnormalized data!
- Example: korpling.german.hu-berlin.de/annis3
  →Ridges corpus (type 'ridges' in 'Filter' (left)
  →Query p="p" finds full sentences
  →Set displayed contexts to 0 ('search options')
  →Explore normalization strategy ('all annotation')
- (Documentation:
  http://korpling.german.hu-berlin.de/ridges/documentation_v5_en.html)

Side note: Processing strategies for non-normalized language data

- Fields that cause NLP problems
  - CMC and web-crawled language data
  - Language acquisition data
  - Non-standard language varieties (dialects, spoken registers, …)
  - Historical texts
- → Problems relatively similar (for a summary, compare Eller&Hirschmann 2014)
- Normalization vs. robustness
  - Robustness: Increasing ability of analyzer to assign correct (=wanted) analysis to non-canonical structure
    - Extending lexicon: adding non-canonical forms
    - Removing restrictions for analysis (German nouns → capital letters)
    - Statistical methods
    - …
  - **Normalization: Making non-canonical structure canonical**

Methods of syntactic annotation: Part of speech (pos)

- Tagset: Complete list of categories for syntactic words
- Basis for further syntactic interpretation
- For syntactic analysis, syntactically motivated pos tags needed
  - *Das*/PRON-DEMSTR *löst*/V *das*/ART-DEF *Problem*/N
    Das/PRON-DEMSTR *löst*/V *manches*/PRON-INDEF *Problem*/N
    Das/PRON-DEMSTR *löst*/V *manches*/ART-INDEF *Problem*/N
- Standard German tagset: STTS (Schiller et al. 1999)
- More fine-grained tagset (inflectional categories): RFTagger (Schmid&Laws 2008)
- Many adaptations of STTS, e.g. DDDTS (http://www.deutschdiachrondigital.de/data/home/manual/dateien/TagSetPoS.pdf)

Methods of syntactic annotation: Part of speech (pos)

- Annotation tools vs. data formats
- Tools:
  - Manual tools such as
    - EXMARaLDA
    - MS Excel
    - Text Editors
    - …
  - Automatic tools (taggers) such as
    - Treetagger (http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/)
    - RFTagger (http://www.cis.uni-muenchen.de/~schmid/tools/RFTagger/)
    - CLAWS tagger (http://ucrel.lancs.ac.uk/claws/)

Methods of syntactic annotation: Part of speech (pos)

- Data formats
  - Inline annotations
    - Das/PDS ist/VAFIN schön/ADJD
    - Das<pos:VAFIN> ist<pos:VAFIN> schön<pos:ADJD>
  - Standoff annotations (separated layers)
    - Treetagger column format

Das	PDS	der
ist	VAFIN	sein
schön	ADJD	schön

    - Standoff xml (e.g. EXMARaLDA xml)

```
<tier id="TIE0" category="text" type="t" display-name="text">
<event start="T0" end="T1">Das</event>
<event start="T1" end="T2">ist</event>
<event start="T2" end="T5">schön</event>
</tier><tier id="TIE1" category="pos" type="a" display-name="pos">
<event start="T0" end="T1">PDS</event>
<event start="T1" end="T2">VAFIN</event>
<event start="T2" end="T5">ADJD</event>
</tier><tier id="TIE2" category="lemma" type="a" display-name="lemma">
<event start="T0" end="T1">der</event>
<event start="T1" end="T2">sein</event>
<event start="T2" end="T5">schön</event>
</tier></basic-body>
```

Methods of manual syntactic annotation: Span annotations

- Representing syntactic categories as spans that cover any number of consecutive tokens
- Syntactic concepts needed that allow for flat representation
  - Simple chunking (maximal or minimal phrases)
  - Topological fields
  - …

Der	Mensch	verfügt	über	die	stark	ausgeprägte	Diskriminationsfähigkeit	,	
d	Mensch	verfügen	über	d	stark	ausgeprägt		,	
ART	NN	VVFIN	APPR	ART	ADJD	ADJA	NN	$,	

⊟ falko (grid)

lemma	d	Mensch	verfügen	über	d	stark	ausgeprägt		,
pos	ART	NN	VVFIN	APPR	ART	ADJD	ADJA	NN	$,
word	Der	Mensch	verfügt	über	die	stark	ausgeprägte	Diskriminationsfähigkeit	,
tok	Der	Mensch	verfügt	über	die	stark	ausgeprägte	Diskriminationsfähigkeit	,

⊟ FalkoSummay2v1 (grid)

konstituenten-satz_1									x
matrix-satz	x								
matrix-satz_felder	VF_MS		LSK_MS	MF_MS					NF_MS
target_hypothesis					eine				
tok	Der	Mensch	verfügt	über	die	stark	ausgeprägte	Diskriminationsfähigkeit	,

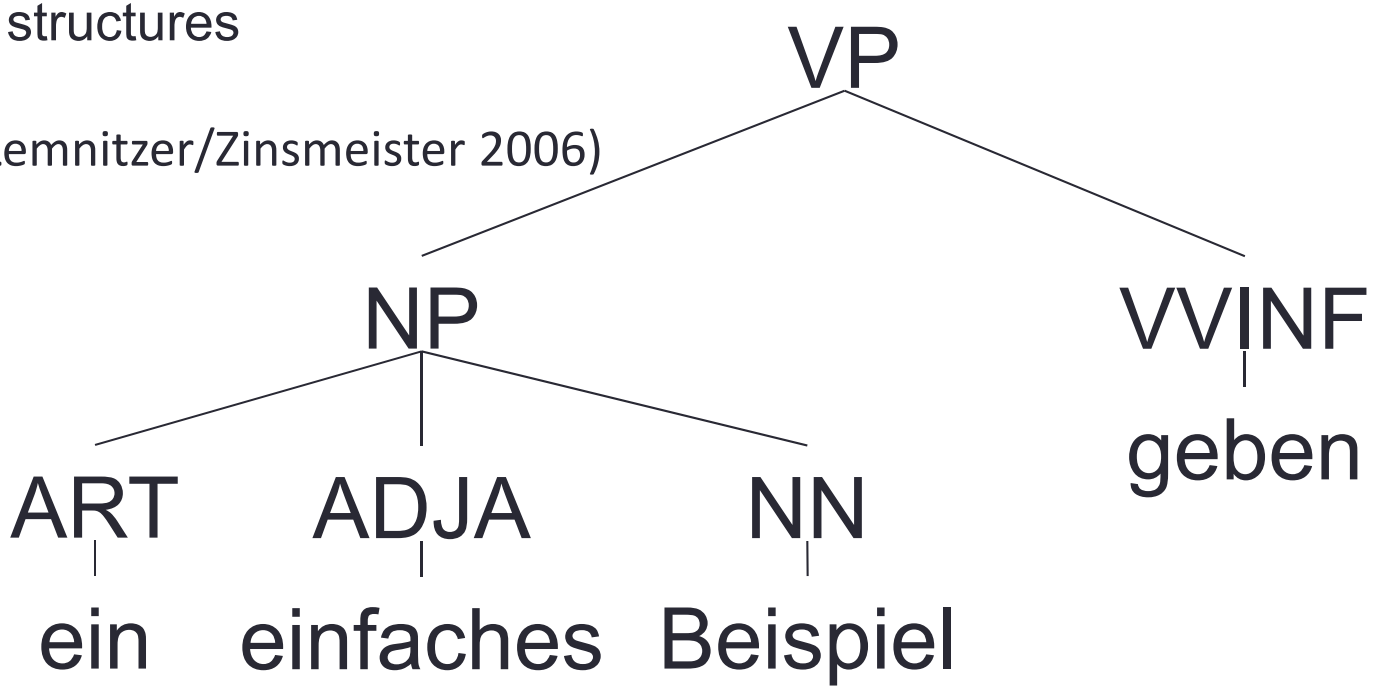
Tools for manual syntactic annotation: Span annotations

- Programs that can cluster any number of consecutive tokens
  - EXMARaLDA (exmaralda.org)
  - ELAN (tla.mpi.nl/tools/tla-tools/elan/)
  - MMAX2 (mmax2.sourceforge.net/)
  - MITRE Annotation Toolkit (MAT) (http://mat-annotation.sourceforge.net/)
  - MS Excel, Open Office Calc, LibreOffice Calc
  - …

Methods of corpus based syntactic annotation: Phrase structures

- Phrase structures
  - →Recursively attaching elementary units (daughter units) to phrases (mother units)
  - →complex hierarchical structures of terminal and non-terminal nodes
  - →Tree structures

(According to Lemnitzer/Zinsmeister 2006)



Phrase structure representation and data format

- Data format needs to represent hierarchical structures of any complexity
  - Span annotations not a suitable format
- xml data formats such as
  - tiger xml (see examples in data set)
  - paula xml
  - Prague Markup Language (pml)
  - …

Tools for manual syntactic annotation: Phrase structures

- Aim: Assign phrase structures to tokenized text data
- Very few modern GUI based programs that can edit and store
  - Annotate (no longer supported)
  - Synpathy (buggy, no longer supported) (https://tla.mpi.nl/tools/tla-tools/older-tools/synpathy/)
  - TrEd (Tree editor, for Prague Dependency Treebank) (https://ufal.mff.cuni.cz/tred/)
  - In preparation: Atomic (http://corpus-tools.org/atomic/)

Tools for automatic syntactic annotation: Phrase structures

- Aim: Automatically assign phrase structures to tokenized text data
  - Programs based on training data (gold standard data)
  - Some provide grammatical functions
  - Different data input formats:
    - Tokenization expected or provided by parser
    - pos annotations already in input data or parser uses own tagger
    - One sentence per line → Preceding sentence segmentation needed
  - Different data output formats
    - TIGER xml
    - Bracketing formats

Tools for automatic syntactic annotation: Phrase structures

- **Stanford parser** (http://nlp.stanford.edu/software/lex-parser.shtml)
  - Input: Takes tokenized textual data with pos tags:
    Sobald/KOUS die/ART Frau/NN
  - Output: .txt with Penn Treebank Bracketing structures:
    (ROOT(NUR(S(S (KOUS Sobald)(NP (ART die) (NN Frau) …
  - Output can be converted into other formats (via TIGER registry tool; http://www.ims.uni-stuttgart.de/data/TIGERSearchTools.zip)
  - Output contains full sentence parses with phrase structure trees and phrasal labels (TIGER scheme)
    - Functional labels (according to TIGER scheme) possible
- **Berkeley parser** (https://github.com/slavpetrov/berkeleyparser)
  - Input:

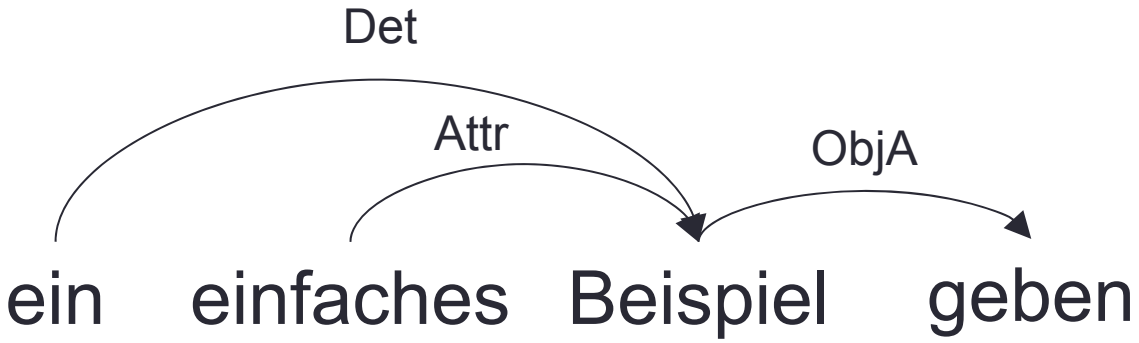
Sobald	KOUS
die	ART
Frau	NN

    (Sentences separated by blank line)
  - Output: Like Stanford parser, but TÜBA-DZ structure format with topological fields

Methods of corpus based syntactic annotation: Dependency structures

- Dependencies
  - →Recursively attaching elementary daughter units to elementary mother units
  - →complex hierarchical structures of terminal nodes
  - →Tree structures

(According to Lemnitzer/Zinsmeister 2006)



Tools for manual syntactic annotation: Dependency structures

- Arborator (http://arborator.ilpga.fr/; http://arborator.ilpga.fr/q.cgi)
  - See and edit CONLL table format and graphical interpretation simultaneously **online** and as server installation
- TrEd (https://ufal.mff.cuni.cz/tred/)
  - Multiple format tree structure editor
  - Can convert between input and export formats

Tools for automatic syntactic annotation: Dependency structures

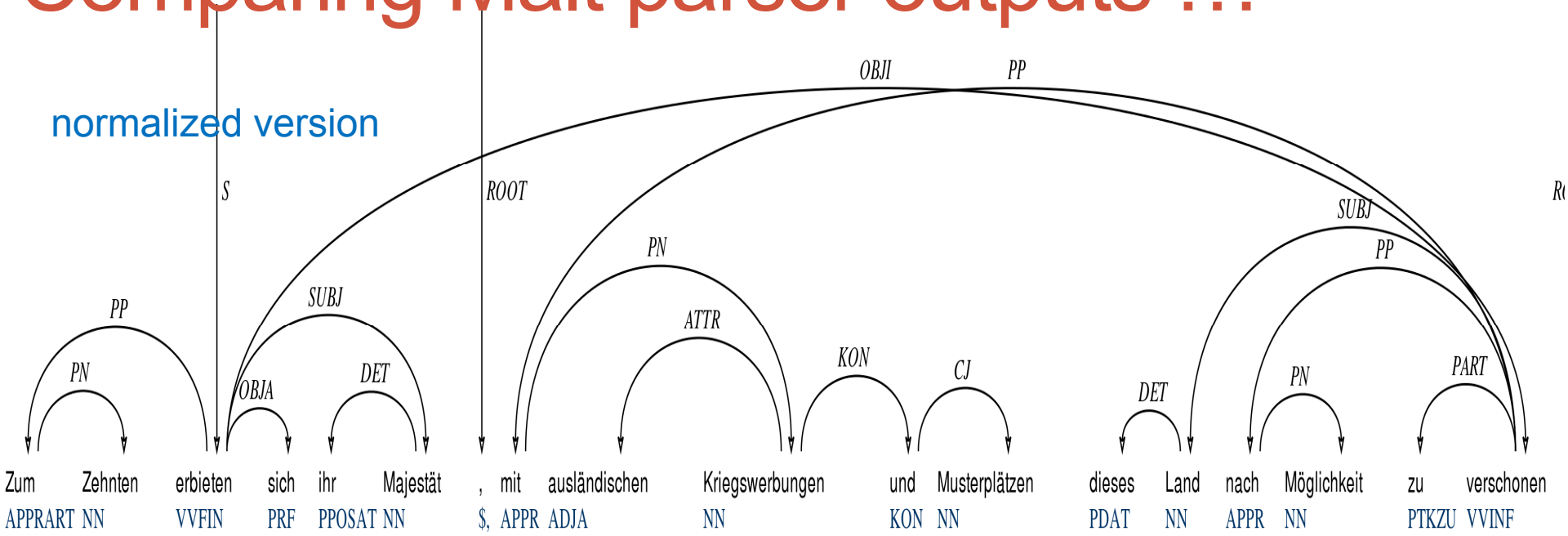
- **Malt parser** (http://www.maltparser.org/)
  - Input: Takes tokenized, pos tagged, sentence separated textual data in table format (CONLL):
  - 1 Sobald -- KOUS KOUS _
    2 die -- ART ART _
    3 Frau -- NN NN _
  - Output: Adds dependency information – ID of depenent token, grammatical function:
  - 1 Sobald -- KOUS KOUS 6 KONJ
    2 die -- ART ART 3 DET
    3 Frau -- NN NN 6 SUBJ
- **Mate tools** (https://github.com/slavpetrov/berkeleyparser)
  - Input: Unannotated but tokenized text, one sentence per line
  - Output: like Malt parser, different tagset

Semi-automatic annotation - The case of ENHG

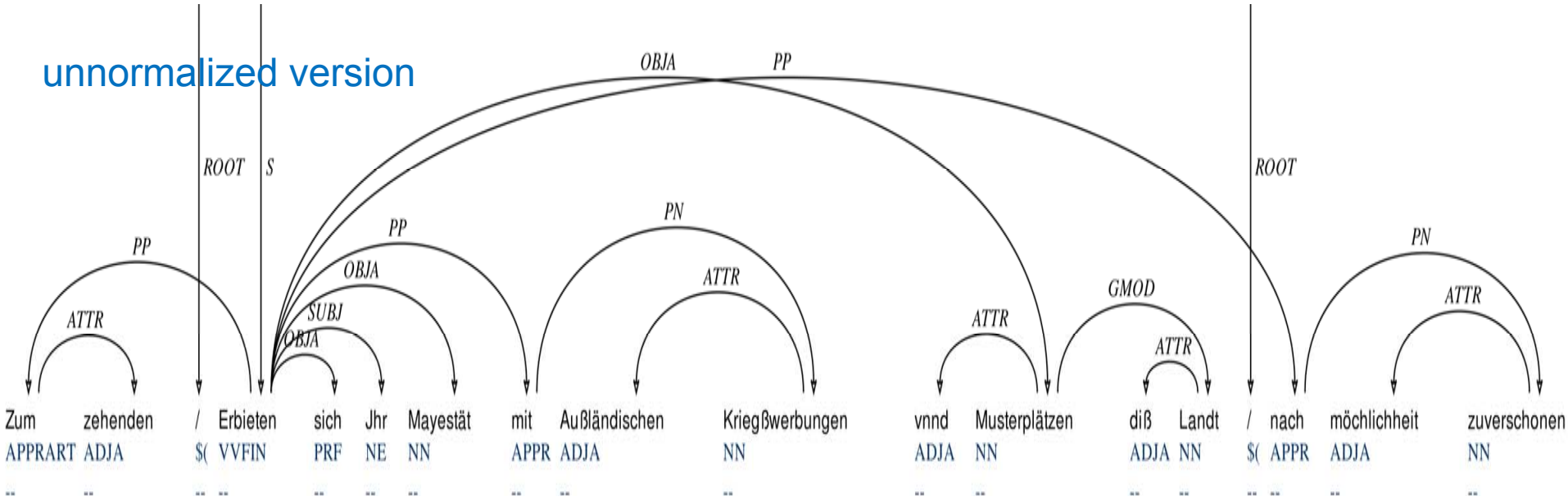
- Aim: Creating parsable input for syntactical parsers
- Example: Data from Mercurius corpus (only one exemplary sentence)
- Normalizing data manually
- Parsing unnormalized data
- Parsing normalized data

Comparing Malt parser outputs …

normalized version

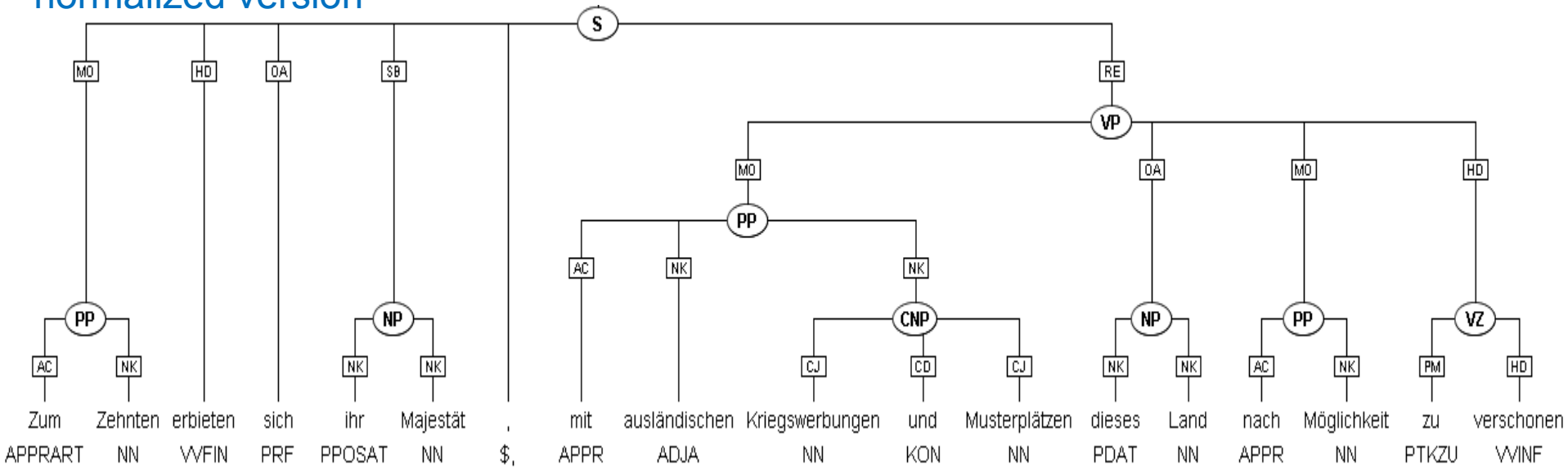


unnormalized version

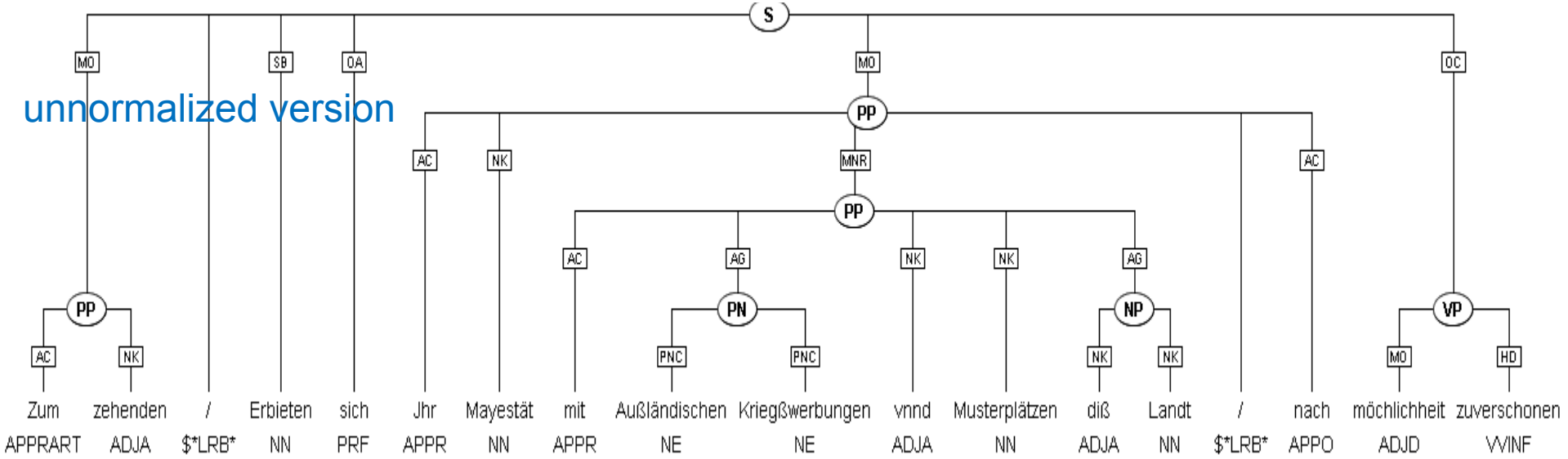


Comparing Berkeley parser outputs …

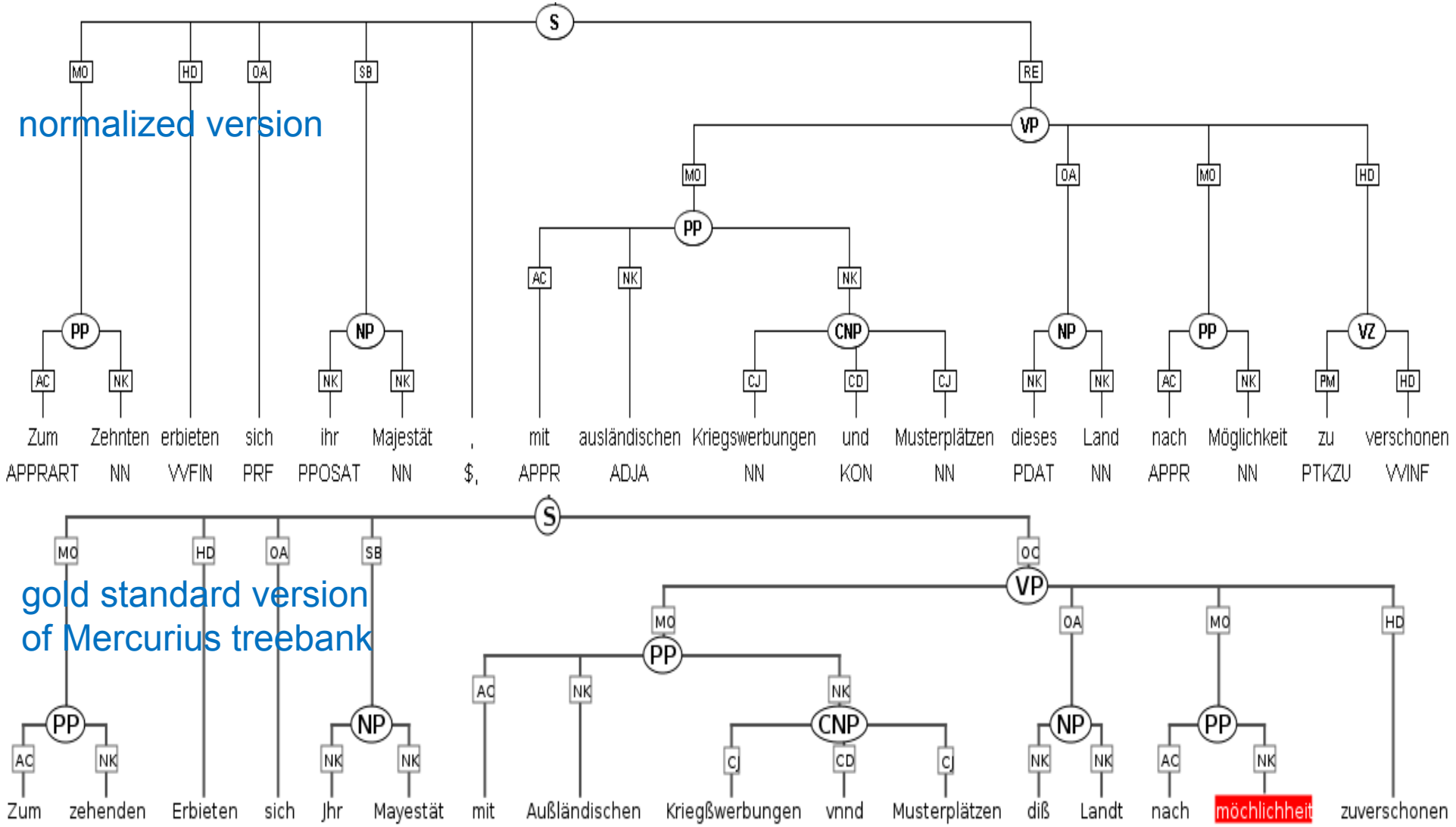
normalized version



unnormalized version



Comparing normalized Berkeley parser output and gold standard …



Editing automatically parsed data (suggestions)

- TIGER xml phrase structures:
  - Install TrEd (portable) (https://ufal.mff.cuni.cz/tred/)
  - Load TIGER xml extension
  - Open converted parser output format (TIGER xml)
  - Edit edges and labels
- Dependency structures:
  - Open Arborator editor webpage: http://arborator.ilpga.fr/q.cgi
  - Insert parser output format (CONLL)
  - Edit edges and labels

Searching in processed corpus data

- Import data into search program
- Specific search programs:
  - TIGER xml output format → TigerSearch (http://www.ims.uni-stuttgart.de/data/TIGERSearchTools.zip)
  - CONLL output format → ICARUS (http://www.ims.uni-stuttgart.de/forschung/ressourcen/werkzeuge/icarus.en.html)
  - EXMARaLDA annotations → EXAKT (http://www.exmaralda.org/en/tool/exakt/)
- Format independent program:
  - ANNIS (http://corpus-tools.org/annis/)
    - Convert and merge all above named formats into ANNIS data format via converter framework Pepper (http://corpus-tools.org/pepper/index.html)
    - Search multiple annotations simultaneously

Searching in tree structures via ANNIS

- Example:
  Mercurius treebank
  (http://www.uni-potsdam.de/guvdds/ressourcen/korpora.html)
  - 2 ENHG Newpapers, annotated in TIGER treebank format
- Mercurius in ANNIS: korpling.german.hu-berlin.de/annis3/
  - Type "Mercurius" in 'Filter' box (left)
  - Click on corpus
  - Explore corpus content via *i*-button (next to selected corpus)

Searching in tree structures via ANNIS

- **Search scenarios/requests**
  - Phrase types: Finding PPs or other phrases
    - PPs at left periphery?
  - Phrases and functions: Finding specific PPs:
    - modifying vs. argument PPs
  - Extracting word lists: Creating list of verbal heads which select specific PP heads as arguments/objects
  - Valency structure: ditransitive constructions
  - A negation word preceding any kind of verb
  - Word order:
    - Adnominal adjectives within noun phrase after noun?
      Comparing ENHG and OHG data
    - Finding finite verbs in subordinate clauses and v2 position

Searching in tree structures via ANNIS

These searches work in Mercurius corpus

- Phrase types: Finding PPs:

  cat="PP"

  →Finds all structures that are interpreted as PPs in the data

  →See TIGER phrase label set for other possible phrase category according to TIGER scheme

- PPs at right periphery of mother structure:

  node >@r cat="PP"

→Finds all PPs that are annotated rightmost in mother structure

- Specify mother phrase: Only top phrase structure in sentence (according to TIGER scheme):

  cat="S" >@r cat="PP"

Searching in tree structures via ANNIS

These searches work in Mercurius corpus

- Phrases and functions: Finding specific PPs:
  - modifying vs. argument PPs:
    - modifying PPs
      node >[label="MO"] cat="PP"

      →Finds every PP that is analyzed as a modifier of mother node
    - PP arguments
      node >[label="OP"] cat="PP"

Searching in tree structures via ANNIS

These searches work in Mercurius corpus

- Valency structure: ditransitive constructions

node >[label="OA"] node &

#1 >[label="DA"] node

→Finds every phrase that has an accusative object and a dative daughter ('#1' refers to first named variable (node that dominates))

→You can specify every 'node' by a phrase category ('cat="XY") according to TIGER phrase label list

→If dominated nodes should be e.g. personal pronouns, replace them by ' pos="PPER" '

Searching in tree structures via ANNIS

These searches work in Mercurius corpus

- A negation word preceding any kind of verb

  pos="PTKNEG" . pos=/V.*/

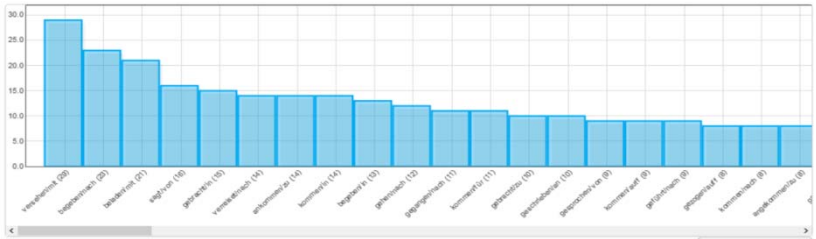
- A negation word that should NOT be "nicht" preceding any kind of verb

  pos="PTKNEG" . pos=/V.*/ &
  #1 _=_ tok!="nicht"

Searching in tree structures via ANNIS

These searches work in Mercurius corpus

- Extracting word lists: Creating list of verbal heads which select specific PP heads as arguments/objects
- Query:
  node >[label="OP"] node &
- #1 >[label="HD"] tok &
- #2 >[label="AC"] tok
- Now use 'More'>'Frequency Analysis', erase unwanted variables (1&2), keep 3&4 and get list



Download as CSV

1071 items with a total sum of 1816 (query on Mercurius)

rank	#3 | spanned text	#4 | spanned text	count
1	versehen	mit	29
2	begeben	nach	23
3	beladen	mit	21

Searching in tree structures via ANNIS

These searches work in Mercurius corpus

- **Finding finite verbs in subordinate clauses but v2 position**
- 1. Sequence 'Subjunction-any element-verbal head-no forward slash' in the same sentence

```
cat="S" >[label="CP"] tok &
#1 >[label="HD"] tok &
#2 .2 #3 &
#3 . tok!="/"
```

- 2. Sequence 'Subjunction-left element of phrase…right element of phrase-verbal head' in the same sentence

```
cat="S" >[label="CP"] tok &
#1 >[label="HD"] tok &
#1 > cat=/(PP|NP|AP|AVP)/ &
#4 >@l tok &
#4 >@r tok &
#2 . #5 &
#6 . #3
```

Searching in tree structures via ANNIS

- **Finding finite verbs in subordinate clauses but v2 position**

Same search goal in RIDGES corpus (Version 5.0)

- In RIDGES, subjunctions (pos="KOUS") in subordinate clauses are annotated according to verb position (variable is Verbposition, categories are Vletzt (verb final), V2 (verb second), and V1 (verb initial).
(These are EXMARaLDA annotations)
→ Search for
pos="KOUS" _=_ Verbposition="V2"
in order to find all annotated occurrences of subjunctions that dominate V2 strucures (_=_ means "x equals y")
→If you want to create of list of subjunctions that dominate V2 in the data, search for
pos="KOUS" _=_ Verbposition="V2" _=_ lemma
use 'More'→'Frequency, delete unwanted variables "pos" and "Verbposition" and create list of lemmas that allow V2

Conclusion

- (Of course) you can only search for categories that are annotated in a given corpus
- (It is always possible to add new annotation layers to a given corpus if you need to do that)

Span annotation scenario (EXMARaLDA annotations)

This search works in DDD-Heiland

- In **DDD-Heiland**, sentence types are annotated as spans over the tokens covering the respective sentence
- Aim: We want to find subjunctions (pos="KOUS") that are contained only in sentences of the type finite, introduced sentence, adverbial clause, modal function (→tag CF_I_Adv_Mod on variable clause)
- →query:

clause="CF_I_Adv_Mod" _i_ pos="KOUS"
(→ _i_ means "x includes y")