Linguistic annotation

From morphology to pragmatics

Dag Haug

University of Oslo

11 September 2023

Table of Contents

Basic questions

- What is linguistic annotation?
- Why would you want it?

The foundation

- In principle, anything that goes beyond a mere picture (or recording) of linguistic data is interpretation/data enrichment/annotation
  - letter interpretations
  - word boundaries
  - sentence boundaries
  - diacritics that aren't in the source
  - ...
- But we think of that as part of editions
- There are a lot of digital editions these days, but for linguistic purposes we will often just pick an edition

Choosing an edition

- How to pick depends on your interests and your material!
  - openly available vs. best/most recent
  - need for philological accuracy
  - parallel data
  - well-preserved vs. fragmentary
  - epigraphic or papyrological material
  - need to document variants?

Three cases

- The PROIEL Greek NT
- DAMOS
- Menota/Menotec

Table of Contents

Morphological annotation

- Appears deceptively simple
- But there are number of issues
  - Finegrainedness of categorization
  - Cross-lingual categories?
  - Disambiguation

Categorization

- What is the appropriate level of granularity?
  - Parts of speech (which?!)
  - Sub-types
  - Morphological categories
- What should determine the category?
  - inherent property vs. contextual usage in e.g. nominalizations? adjectives used as adverbs?
  - cross-lingual considerations (nominative means pretty much the same across languages, gerund less so)

Disambiguation

- A completely lack of diambiguation makes the annotation less useful
- But what belongs in the morphology and what in the syntax?
- Can too much disambiguation also be a problem?

The parts of speech in PROIEL

- Verb
- Noun
- Adjective
- Numeral
- Article
- Pronoun
- Adverb
- Preposition
- Conjunction
- Subjunction
- Conjunction
- Interjection
- Foreign word

Finegrained POS in PROIEL

- **Noun**
  - proper/common
- **Numeral**
  - cardinal
  - ordinal
- **Adverb**
  - relative
  - interrogative
  - other
- **Pronoun**
  - relative
  - interrogative
  - indefinite
  - demonstrative
  - personal
  - possessive
  - personal reflexive
  - possessive reflexive
  - reciprocal

Feature values

feature	possible values
person	1, 2, 3

Feature values

feature	possible values
person	1, 2, 3
number	s (singular), d (dual), p (plural)

Feature values

feature	possible values
person	1, 2, 3
number	s (singular), d (dual), p (plural)
tense	l (pluperfect), a (aorist), p (present), f (future), r (perfect), s (resultative), i (imperfect), t (future perfect), u (past)

Feature values

feature	possible values
person	1, 2, 3
number	s (singular), d (dual), p (plural)
tense	l (pluperfect), a (aorist), p (present), f (future), r (perfect), s (resultative), i (imperfect), t (future perfect), u (past)
mood	m (imperative), n (infintive), o (optative), d (gerund), p (participle), g (gerundive), s (subjunctive), i (i ndicative), u (supine)

Feature values

feature	possible values
person	1, 2, 3
number	s (singular), d (dual), p (plural)
tense	l (pluperfect), a (aorist), p (present), f (future), r (perfect), s (resultative), i (imperfect), t (future perfect), u (past)
mood	m (imperative), n (infintive), o (optative), d (gerund), p (participle), g (gerundive), s (subjunctive), i (i ndicative), u (supine)
voice	a (active), m (middle), e (middle or passive), p (passive)

Feature values

feature	possible values
person	1, 2, 3
number	s (singular), d (dual), p (plural)
tense	l (pluperfect), a (aorist), p (present), f (future), r (perfect), s (resultative), i (imperfect), t (future perfect), u (past)
mood	m (imperative), n (infintive), o (optative), d (gerund), p (participle), g (gerundive), s (subjunctive), i (i ndicative), u (supine)
voice	a (active), m (middle), e (middle or passive), p (passive)
gender	m (masculine), f (feminine), n (neuter), o (m. or n.), p (m. or f.), q (m., f. or n.), r (f. or n.)

Feature values

feature	possible values
person	1, 2, 3
number	s (singular), d (dual), p (plural)
tense	l (pluperfect), a (aorist), p (present), f (future), r (perfect), s (resultative), i (imperfect), t (future perfect), u (past)
mood	m (imperative), n (infintive), o (optative), d (gerund), p (participle), g (gerundive), s (subjunctive), i (i ndicative), u (supine)
voice	a (active), m (middle), e (middle or passive), p (passive)
gender	m (masculine), f (feminine), n (neuter), o (m. or n.), p (m. or f.), q (m., f. or n.), r (f. or n.)
case	n (nominative), a (accusative), g (genitive), d (dative), c (genitive or dative), i (instrumental), b (ablative), l (locative)

Feature values

feature	possible values
person	1, 2, 3
number	s (singular), d (dual), p (plural)
tense	l (pluperfect), a (aorist), p (present), f (future), r (perfect), s (resultative), i (imperfect), t (future perfect), u (past)
mood	m (imperative), n (infintive), o (optative), d (gerund), p (participle), g (gerundive), s (subjunctive), i (i ndicative), u (supine)
voice	a (active), m (middle), e (middle or passive), p (passive)
gender	m (masculine), f (feminine), n (neuter), o (m. or n.), p (m. or f.), q (m., f. or n.), r (f. or n.)
case	n (nominative), a (accusative), g (genitive), d (dative), c (genitive or dative), i (instrumental), b (ablative), l (locative)
degree	p (positive), c (comparative), s (superlative)

Feature values

feature	possible values
person	1, 2, 3
number	s (singular), d (dual), p (plural)
tense	l (pluperfect), a (aorist), p (present), f (future), r (perfect), s (resultative), i (imperfect), t (future perfect), u (past)
mood	m (imperative), n (infintive), o (optative), d (gerund), p (participle), g (gerundive), s (subjunctive), i (i ndicative), u (supine)
voice	a (active), m (middle), e (middle or passive), p (passive)
gender	m (masculine), f (feminine), n (neuter), o (m. or n.), p (m. or f.), q (m., f. or n.), r (f. or n.)
case	n (nominative), a (accusative), g (genitive), d (dative), c (genitive or dative), i (instrumental), b (ablative), l (locative)
degree	p (positive), c (comparative), s (superlative)
strength	w (weak), s (strong), t (weak or strong)

Feature values

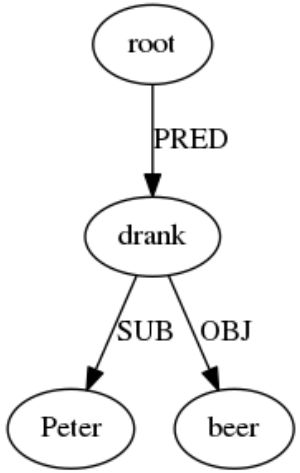
feature	possible values
person	1, 2, 3
number	s (singular), d (dual), p (plural)
tense	l (pluperfect), a (aorist), p (present), f (future), r (perfect), s (resultative), i (imperfect), t (future perfect), u (past)
mood	m (imperative), n (infintive), o (optative), d (gerund), p (participle), g (gerundive), s (subjunctive), i (i ndicative), u (supine)
voice	a (active), m (middle), e (middle or passive), p (passive)
gender	m (masculine), f (feminine), n (neuter), o (m. or n.), p (m. or f.), q (m., f. or n.), r (f. or n.)
case	n (nominative), a (accusative), g (genitive), d (dative), c (genitive or dative), i (instrumental), b (ablative), l (locative)
degree	p (positive), c (comparative), s (superlative)
strength	w (weak), s (strong), t (weak or strong)
inflection	i (inflecting), n (non-inflecting)

Comparing corpora

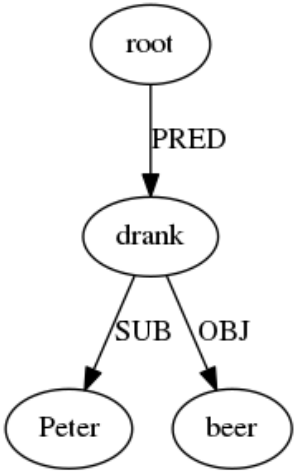
- PROIEL and Menotec have relatively similar categories – why?
- What about morphological annoation in DAMOS? Does not exist, but what would it look like?

Table of Contents

Dependency Grammar

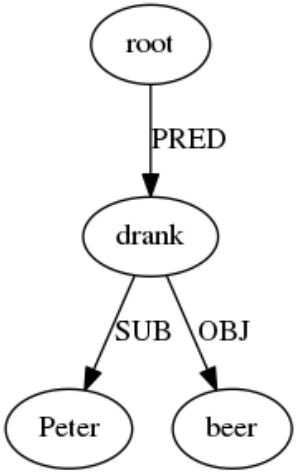


Dependency Grammar



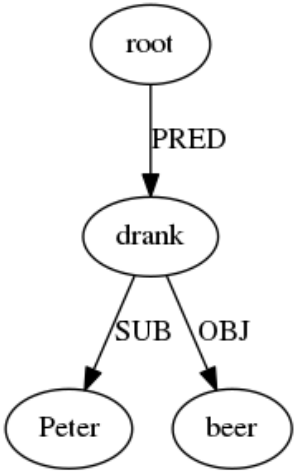
- Dependencies are asymmetric relations between words

Dependency Grammar



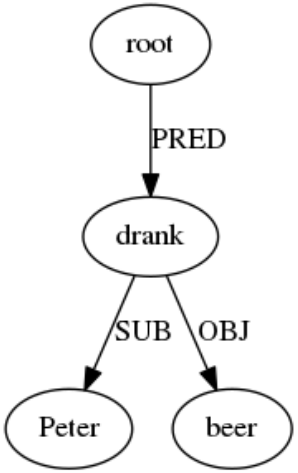
- Dependencies are asymmetric relations between words
- We label these dependencies with the function of the dependent

Dependency Grammar



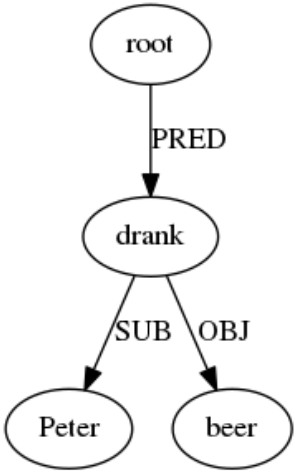
- Dependencies are asymmetric relations between words
- We label these dependencies with the function of the dependent
- The dependencies form a tree under an abstract root

Dependency Grammar



- Dependencies are asymmetric relations between words
- We label these dependencies with the function of the dependent
- The dependencies form a tree under an abstract root
- No explicit constituency, separation of syntactic relations and word order

Dependency Grammar



- Dependencies are asymmetric relations between words
- We label these dependencies with the function of the dependent
- The dependencies form a tree under an abstract root
- No explicit constituency, separation of syntactic relations and word order

Dependency Grammar (continued)

- Almost a standard in computational linguistics
  - computationally simple
  - good parsers available

Dependency Grammar (continued)

- Almost a standard in computational linguistics
  - computationally simple
  - good parsers available
- A natural choice for linguistic databases
  - fairly close to traditional school grammar $\rightarrow$ easy to annotate
  - relatively 'theory-neutral'
  - taking traditional grammatical relations as primitives
- However, there are certainly debates within the dependency community!

Storing linguistic analyses

- TTheory-neutrality $\rightarrow$
  - data for larger audiences

Storing linguistic analyses

- TTheory-neutrality $\rightarrow$
  - data for larger audiences
  - widening gulf between corpus linguistics and linguistic theory

Storing linguistic analyses

- TTheory-neutrality $\rightarrow$
  - data for larger audiences
  - widening gulf between corpus linguistics and linguistic theory
- DG corpora (Prague, PROIEL) $\rightarrow$ DG not really in use as a linguistic theory

Storing linguistic analyses

- TTheory-neutrality $\rightarrow$
  - data for larger audiences
  - widening gulf between corpus linguistics and linguistic theory
- DG corpora (Prague, PROIEL) $\rightarrow$ DG not really in use as a linguistic theory
- PS corpora (Penn, NEGRA) typically use flatter tree structures than anyone believes in

Storing linguistic analyses

- TTheory-neutrality $\rightarrow$
  - data for larger audiences
  - widening gulf between corpus linguistics and linguistic theory
- DG corpora (Prague, PROIEL) $\rightarrow$ DG not really in use as a linguistic theory
- PS corpora (Penn, NEGRA) typically use flatter tree structures than anyone believes in
- On the other hand, LFG and HPSG corpora can be hard to use for people who do not share the theoretical assumptions of these theories

Our take

Principles

1. Encode no more structure than is common to all frameworks

Our take

Principles

1. Encode no more structure than is common to all frameworks
2. Enoded structure could be seen as derived/secondary in some frameworks

Our take

Principles

1. Encode no more structure than is common to all frameworks
2. Enoded structure could be seen as derived/secondary in some frameworks
3. Encode enough structure to allow reconstruction of theoretically motived structures

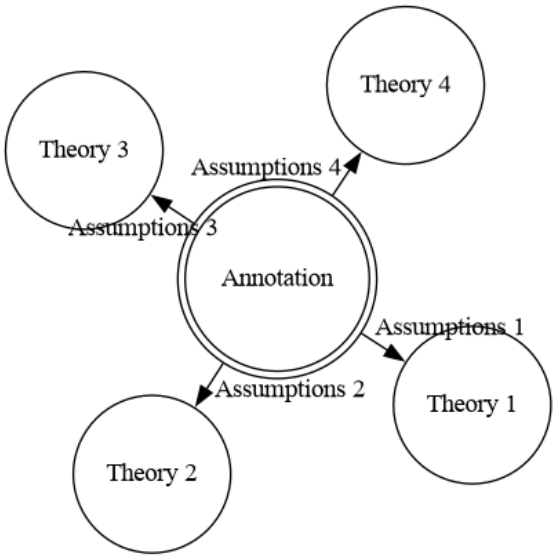
Our take

Principles

1. Encode no more structure than is common to all frameworks
2. Enoded structure could be seen as derived/secondary in some frameworks
3. Encode enough structure to allow reconstruction of theoretically motived structures

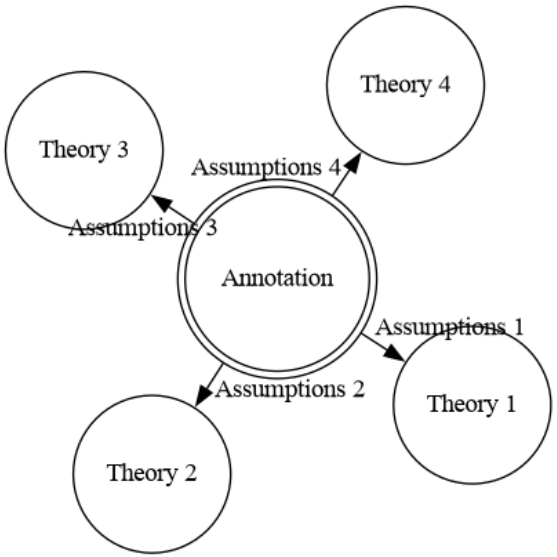
- In the ideal situation, the information in the annotation can be (monotonically) expanded to structures conforming to a particular theory by adding information from the assumptions of that theory

The ideal situation



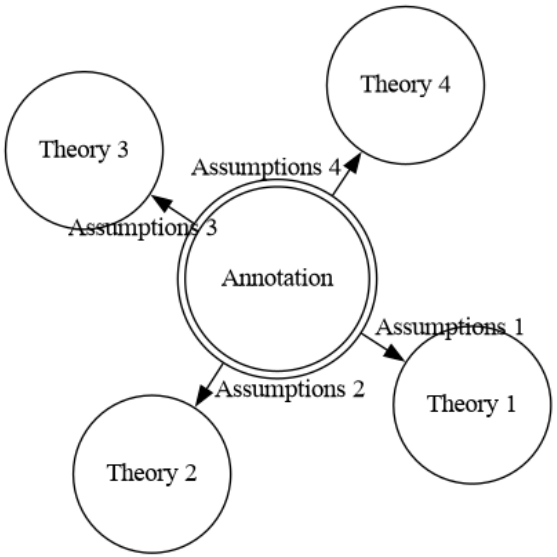
- The added assumptions will typically be about phrase structure, such as various versions of X′ theory

The ideal situation



- The added assumptions will typically be about phrase structure, such as various versions of X′ theory
- Given information about what the subject is, it will be possible to create a structure where the subject has a specific position if the theory requires that (unless the data contradict the theory)

The ideal situation



- The added assumptions will typically be about phrase structure, such as various versions of X′ theory
- Given information about what the subject is, it will be possible to create a structure where the subject has a specific position if the theory requires that (unless the data contradict the theory)
- Useful for testing theories

PROIEL relations

Label	**Function**
PRED	Predicate
SUB	Subject
OBJ	Object
OBL	Oblique
AG	Agent
ADV	Adverbial
ATR	Attribute
APOS	Apposition
NARG	Nominal argument

Label	**Function**
XADV	Free predicative
XOBJ	Open complement
Aux	Auxiliary
XOBJ	Open complement clause
COMP	Complement clause
PART	Partitive
PARPRED	Parenthetical
VOC	Vocative

▸ example ▸ guidelines

Structure sharing

- Subject control: ▸ Example

Structure sharing

- Subject control: ▸ Example
- Object control: ▸ Example

Structure sharing

- Subject control: ▸ Example
- Object control: ▸ Example
- Various other possibilities

Structure sharing

- Subject control: ▸ Example
- Object control: ▸ Example
- Various other possibilities
- Could also be encoded in the label but typically not with the same precision

Structure sharing

- Subject control: ▸ Example
- Object control: ▸ Example
- Various other possibilities
- Could also be encoded in the label but typically not with the same precision
- Not processable by standard dependency parsers, so must be ignored/inserted separately

Differences PROIEL vs. UD

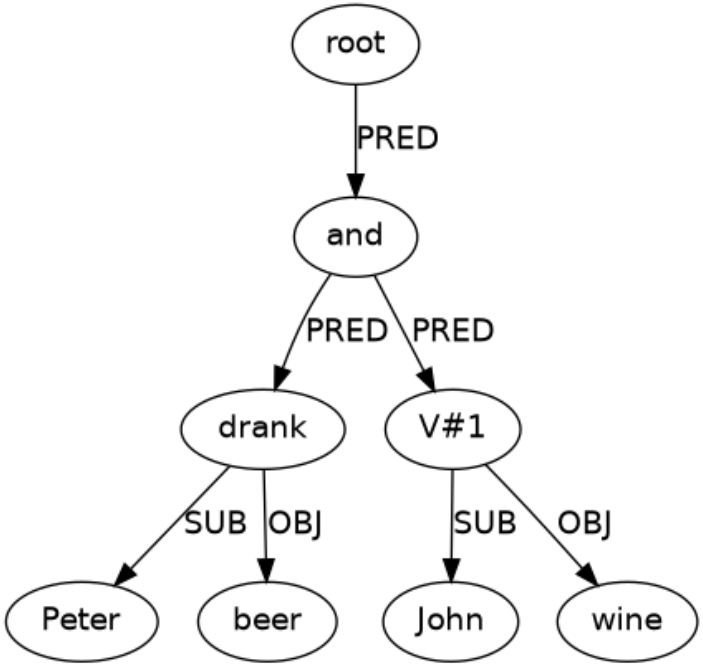
▸ UD relations

- Relation differences
  - No argument–adjunct distinction, only core vs. non-core
  - Attention paid to various "non-linguistic" relations
  - Dedicated relations for coordination
- Structure differences
  - PROIEL has prepositions and complementizers as heads, though determiners and auxiliary verbs as dependents
  - UD has a consistent policy of lexical heads
  - In copular structures, PROIEL takes the copula as head, UD the predicate
  - In coordinate structures, PROIEL takes the conjunction as head, UD the first conjunct

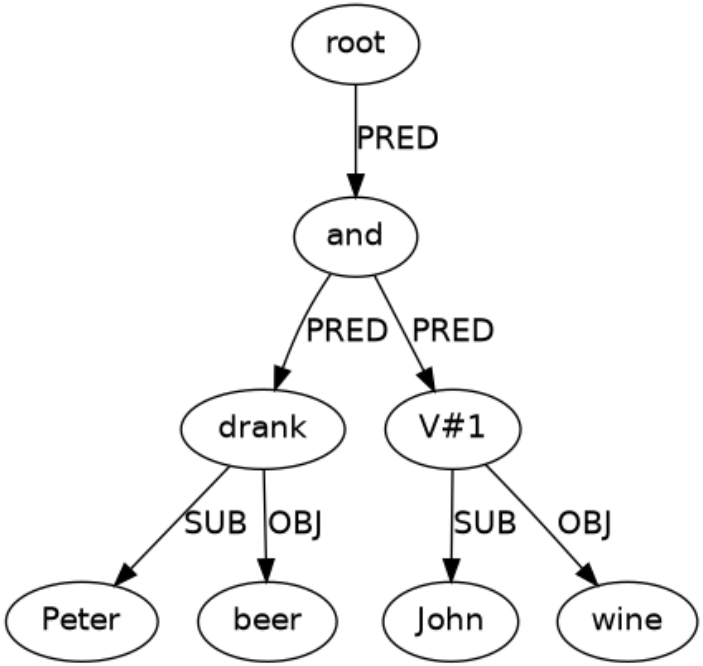
Empty nodes in PROIEL

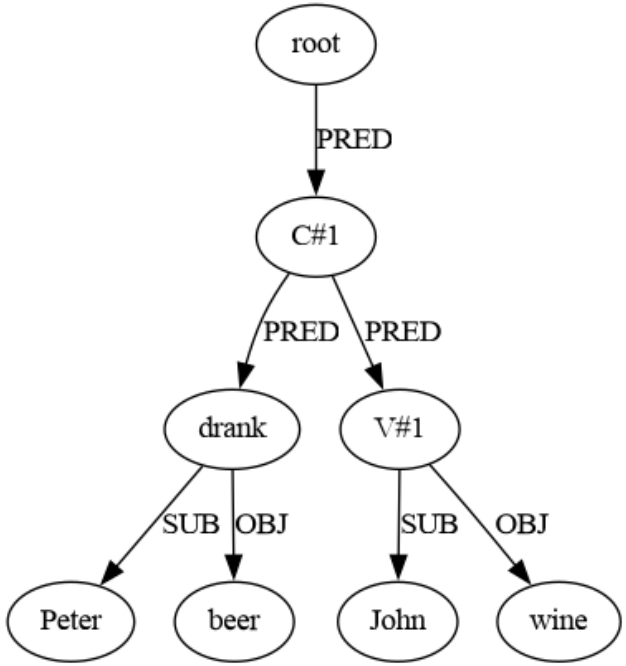
- Only for nodes with dependents:
  - Null V for elided verbs: *Peter drank beer and John wine*
  - Null C for asyndetic parataxis: *citius, altius, fortius*
- The first is pervasive in treebanks, especially of old Indo-European
- The second is only an issue when you have conjunctions as heads

Examples of empty nodes



Examples of empty nodes





Empty nodes example

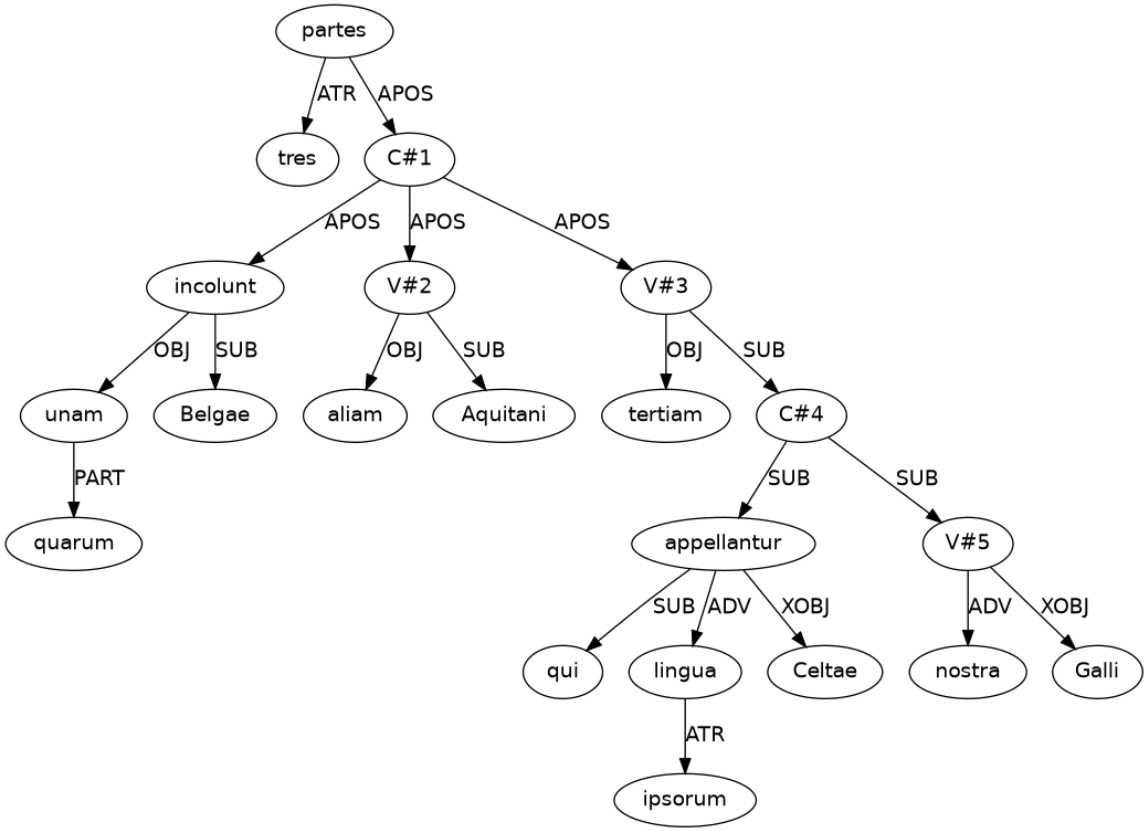
How many empty nodes?

Gallia est omnis divisa in partes tres, quorum unam incolunt Belgae, aliam Aquitani, tertiam qui ipsorum lingua Celtae, nostra Galli appellantur

Empty nodes example

How many empty nodes?

Gallia est omnis divisa in partes tres, quorum unam incolunt Belgae, aliam Aquitani, tertiam qui ipsorum lingua Celtae, nostra Galli appellantur



Exercises

Download the files from `https://daghaug.github.io/`

1. Annotate (together) a sentence from Minucius Felix: `http://foni.uio.no/proiel/sentences/195611`
2. To annotate it in UD, upload `minucius.conllu` in Annotatrix: `https://maryszmary.github.io/ud-annotatrix/standalone/annotator.html`
3. Now annotate the first sentence of the Gallic War (`BGfirst.conllu`) in Annotatrix.
4. Discuss differences and similarities between PROIEL and UD
5. In groups: annotate selected, (relatively) short sentences from the first chapters of the Gallic War
   - In UD, using `BG1_nosyntax.conllu`
   - In PROIEL, using the text Caesar Sandbox 2 and 3 (log in with user `tester1` or `tester2` and password `lakecomo`)
6. Compare with gold standard UD and PROIEL and discuss

Table of Contents

Information structure annotation

- Was a core interest in the PROIEL project
- We concentrated on categories where it is possible to achieve intersubjective agreement
- Mainly givenness and anaphoricity
- Choice of referential expression $\rightarrow$ reference-based notion of givenness
- But there are clear effects of word repetition that we then miss
- And we do not want the choice of referential expression to influence the annotation, to avoid circularity

Mark 14:66-72

66 And as Peter was beneath in the palace, there cometh one of the maids of the high priest: 67 And when she saw Peter warming himself, she looked upon him, and said, And thou also wast with Jesus of Nazareth. 68 But he denied, saying, I know not, neither understand I what thou sayest. And he went out into the porch; and the cock crew. 69 And a maid saw him again, and began to say to them that stood by, This is one of them. 70 And he denied it again. And a little after, they that stood by said again to Peter, Surely thou art one of them: for thou art a Galilaean, and thy speech agreeth thereto. 71 But he began to curse and to swear, saying, I know not this man of whom ye speak. 72 And the second time the cock crew. And Peter called to mind the word that Jesus said unto him, Before the cock crow twice, thou shalt deny me thrice. And when he thought thereon, he wept.

- Greek uses the indefinite here!

Non-standard article usage

Mark 4:3

Hearken; Behold, there went out a sower to sow:

- This is definite in Greek, despite the presentation construction
- Generally freer use of definites in the parables

Quantificational article use

Luke 16:18

(1) *ho apolelumenên apo andros gamôn moikheuei*
the divorced from man.GEN marrying.NOM commit adultery
'Whoever marries a woman divorced from a man commits adultery'

Combining layers

- Some schemes incorporate distinction such as the following in the GIVEN-category (from Riester et al.)
  - -PRONOUN
  - -REFLEXIVE
  - -SHORT
  - -REPEATED
  - -EPITHET
- In PROIEL, information about the type of anaphoric expression is available from the syntactic annotation
- Preferrable, because it gives much more detail, but of course time-consuming if your only interest is information status

Givenness

Givenness — our approach

How can the hearer establish the referent of an NP?

Givenness

Givenness — our approach

How can the hearer establish the referent of an NP?

- Primary question: does the NP have one or more referents at all?

Discourse referents

The heuristics

Does the NP establish a referent that could be picked up in the following discourse?

Discourse referents

The heuristics

Does the NP establish a referent that could be picked up in the following discourse?

1. Jesus saw a leper. He was blind.

Discourse referents

The heuristics

Does the NP establish a referent that could be picked up in the following discourse?

1. Jesus saw a leper. He was blind.
2. Jesus saw some lepers. They were blind.

Discourse referents

The heuristics

Does the NP establish a referent that could be picked up in the following discourse?

1. Jesus saw a leper. He was blind.
2. Jesus saw some lepers. They were blind.
3. Jesus wandered in deserted places. *They were called Sodoma and Gomorra

Discourse referents

The heuristics

Does the NP establish a referent that could be picked up in the following discourse?

1. Jesus saw a leper. He was blind.
2. Jesus saw some lepers. They were blind.
3. Jesus wandered in deserted places. *They were called Sodoma and Gomorra

- In Latin and (and to some extent Greek) we cannot rely on the presence of an overt determiner to decide which NPs are referential

NPs without determiners

Luke 15:14

(2) *egeneto* *limos* *iskhura* *kata tên khôran ekeinên*
become.AOR.3SG famine powerful over the country that
There was (a) great famine over that country

NPs without determiners

Caesar BG 6.35.1

(3) *Transeunt Rhenum navibus ratibus=que*
cross Rhine ships barks=and
They cross the Rhine in (the/their) ships and barks

NPs without determiners

Luke 5:18

(4) *et ecce viri portantes in lecto hominem*
and behold men carrying in bed man

NPs without determiners

Luke 5:18

(4) *et ecce viri portantes in lecto hominem*
and behold men carrying in bed man

Und, siehe, etliche Männer brachten einen Menschen auf seinem Bette

And behold, some men were bringing on a bed a man (English Standard Version)

And behold, men brought in a bed a man (King James)

Determinerless languages

- The English indefinite plural is not the plural of the indefinite and does not normally introduce a discourse referent

Determinerless languages

- The English indefinite plural is not the plural of the indefinite and does not normally introduce a discourse referent
- But in Greek and Latin, the determinerless plural *can but does not have to* introduce a (plural) discourse referent

Determinerless languages

- The English indefinite plural is not the plural of the indefinite and does not normally introduce a discourse referent
- But in Greek and Latin, the determinerless plural *can but does not have to* introduce a (plural) discourse referent
- We need heuristics

Determinerless languages

- The English indefinite plural is not the plural of the indefinite and does not normally introduce a discourse referent
- But in Greek and Latin, the determinerless plural *can but does not have to* introduce a (plural) discourse referent
- We need heuristics
- We assume that singular count nounts introduce DRs, but not mass nouns and plurals – unless they are explicitly picked up in the contexts

Modal subordination

Luke 15:4–5

4 What man of you, having an hundred sheep, if he lose one of them,

Modal subordination

Luke 15:4–5

4 What man of you, having an hundred sheep, if he lose one of them, doth not leave the ninety and nine in the wilderness, and go after that which is lost

Modal subordination

Luke 15:4–5

4 What man of you, having an hundred sheep, if he lose one of them, doth not leave the ninety and nine in the wilderness, and go after that which is lost until he find it?

Modal subordination

Luke 15:4–5

4 What man of you, having an hundred sheep, if he lose one of them, doth not leave the ninety and nine in the wilderness, and go after that which is lost until he find it? 5 And when he hath found it, he layeth it on his shoulders, rejoicing.

Modal subordination

Luke 15:4–5

4 What man of you, having an hundred sheep, if he lose one of them, doth not leave the ninety and nine in the wilderness, and go after that which is lost until he find it? 5 And when he hath found it, he layeth it on his shoulders, rejoicing.

- Quite common in the NT

Modal subordination

Luke 15:4–5

4 What man of you, having an hundred sheep, if he lose one of them, doth not leave the ninety and nine in the wilderness, and go after that which is lost until he find it? 5 And when he hath found it, he layeth it on his shoulders, rejoicing.

- Quite common in the NT
- So we want to deal with embedded/non-specific discourse referents

Modal subordination

Luke 15:4–5

4 What man of you, having an hundred sheep, if he lose one of them, doth not leave the ninety and nine in the wilderness, and go after that which is lost until he find it? 5 And when he hath found it, he layeth it on his shoulders, rejoicing.

- Quite common in the NT
- So we want to deal with embedded/non-specific discourse referents
- But we don't want to mix them with other discourse referents

Modal subordination

Luke 15:4–5

4 What man of you, having an hundred sheep, if he lose one of them, doth not leave the ninety and nine in the wilderness, and go after that which is lost until he find it? 5 And when he hath found it, he layeth it on his shoulders, rejoicing.

- Quite common in the NT
- So we want to deal with embedded/non-specific discourse referents
- But we don't want to mix them with other discourse referents
- We will get back to these

Discourse referents

- Our definition includes most NPs (also those embedded in other NPs!), since even non-specifics can be picked up inside embeddings

Discourse referents

- Our definition includes most NPs (also those embedded in other NPs!), since even non-specifics can be picked up inside embeddings
- But some are excluded:
  - Idiom chunks (*khreian ekhousin* lit. 'have need')
  - NPs in temporal expressions (*he stayed four days*)
  - Appositions (*Herod the king*)

Discourse referents

Null arguments

eipen

say.IND.AOR.3SG

- Greek allows prodrop of subject and non-subject arguments

Discourse referents

Null arguments

eipen
say.IND.AOR.3SG

- Greek allows prodrop of subject and non-subject arguments
- Subject prodrops are easy: just insert them whenever a verb that is not impersonal lacks an overt subjects

Discourse referents

Null arguments

eipen

say.IND.AOR.3SG

- Greek allows prodrop of subject and non-subject arguments
- Subject prodrops are easy: just insert them whenever a verb that is not impersonal lacks an overt subjects
- This example clearly includes a pro subject, but is there a pro dative?

Discourse referents

Null arguments

eipen
say.IND.AOR.3SG

- Greek allows prodrop of subject and non-subject arguments
- Subject prodrops are easy: just insert them whenever a verb that is not impersonal lacks an overt subjects
- This example clearly includes a pro subject, but is there a pro dative?
- We have been restrictive with non-subject null arguments

Null objects

(5) *kai eurôn* ∅ *epitithêsin* ∅ *epi tous ômous autou*
and finding puts on the shoulders his
Und wenn er's gefunden hat, so legt er's auf seine Achseln (Luke 15:5)

- Clear non-subject null arguments that must be inserted

Null objects

(5) *kai eurôn* ∅ *epitithêsin* ∅ *epi tous ômous autou*
and finding puts on the shoulders his
Und wenn er's gefunden hat, so legt er's auf seine Achseln (Luke 15:5)

- Clear non-subject null arguments that must be inserted
- Lacking good valency dictionaries this is impressionistic at the moment

Null objects

(5) *kai eurôn* ∅ *epitithêsin* ∅ *epi tous ômous autou*
and finding puts on the shoulders his
Und wenn er's gefunden hat, so legt er's auf seine Achseln (Luke 15:5)

- Clear non-subject null arguments that must be inserted
- Lacking good valency dictionaries this is impressionistic at the moment
- Can hopefully be made uniform once the corpus is big enough to create a valency dictionary

Meaning contexts

- Givenness tags based on which context the hearer uses to establish reference
  - Discourse (anaphora) $\rightarrow$ OLD

Meaning contexts

- Givenness tags based on which context the hearer uses to establish reference
  - Discourse (anaphora) $\rightarrow$ OLD
  - Situation (deixis) $\rightarrow$ ACC-SIT

Meaning contexts

- Givenness tags based on which context the hearer uses to establish reference
  - Discourse (anaphora) → OLD
  - Situation (deixis) → ACC-SIT
  - Scenarios (inferences) → ACC-INF

Meaning contexts

- Givenness tags based on which context the hearer uses to establish reference
  - Discourse (anaphora) $\rightarrow$ OLD
  - Situation (deixis) $\rightarrow$ ACC-SIT
  - Scenarios (inferences) $\rightarrow$ ACC-INF
  - Encyclopedic knowledge $\rightarrow$ ACC-GEN

Meaning contexts

- Givenness tags based on which context the hearer uses to establish reference
  - Discourse (anaphora) → OLD
  - Situation (deixis) → ACC-SIT
  - Scenarios (inferences) → ACC-INF
  - Encyclopedic knowledge → ACC-GEN
  - No context (no extra-NP information) → NEW

Discourse context

- This is the most straightforward tag
- It is also the most common tag – 58.5% of referents
- It is accompanied by an anaphoric link pointed to the antecedent
- In the case of split antecedents we point to the last one

Situation context

- Another straightforward tag, mostly found in direct speech

Situation context

- Another straightforward tag, mostly found in direct speech
- Rare in our texts – 1.5%

Situation context

- Another straightforward tag, mostly found in direct speech
- Rare in our texts – 1.5%
- Mostly given off by explicit linguistic cues such as demonstratives

Situation context

- Another straightforward tag, mostly found in direct speech
- Rare in our texts – 1.5%
- Mostly given off by explicit linguistic cues such as demonstratives
- Can occur within direct speech with a discourse antecedent outside the direct speech

Situation context

- Another straightforward tag, mostly found in direct speech
- Rare in our texts – 1.5%
- Mostly given off by explicit linguistic cues such as demonstratives
- Can occur within direct speech with a discourse antecedent outside the direct speech
- We mark coreference in these cases

Scenario knowledge

Caesar BG 1.10

(6) *tres quae circum Aquileiam hiemabant ex*
three who around Aquieia winter.IMPF.3P from
hibernis PRO-SUB educit
winter camp.ABL PRO lead
'He led the three legions that were wintering around Aquileia out of (their) winter camps'

Scenario knowledge

Caesar BG 1.10

(6) *tres quae circum Aquileiam hiemabant ex*
three who around Aquieia winter.IMPF.3P from
hibernis PRO-SUB educit
winter camp.ABL PRO lead
'He led the three legions that were wintering around Aquileia out of (their) winter camps'

- Legions → camps

Scenario knowledge

Caesar BG 1.10

(6) *tres quae circum Aquileiam hiemabant ex*
three who around Aquieia winter.IMPF.3P from
hibernis PRO-SUB educit
winter camp.ABL PRO lead
'He led the three legions that were wintering around Aquileia out of (their) winter camps'

- Legions → camps
- When Jesus enters a town, there will typically be lepers around

Scenario knowledge

Caesar BG 1.10

(6) *tres quae circum Aquileiam hiemabant ex*
three who around Aquieia winter.IMPF.3P from
hibernis PRO-SUB educit
winter camp.ABL PRO lead
'He led the three legions that were wintering around Aquileia out of (their) winter camps'

- Legions → camps
- When Jesus enters a town, there will typically be lepers around
- But where to stop?

Restricting scenario knowledge

- We require that the inference is based on lexical evidence, not typical plots
  - The connection should be obvious and stereotypical: example
  - But not necessarily predictable in advance: example

Restricting scenario knowledge

- We require that the inference is based on lexical evidence, not typical plots
  - The connection should be obvious and stereotypical: example
  - But not necessarily predictable in advance: example
- 4.4% of tags

Encyclopaedic context

- These are things that we expect a Hellenized Jew (NT) or an educated Roman (Caesar) to know about

Encyclopaedic context

- These are things that we expect a Hellenized Jew (NT) or an educated Roman (Caesar) to know about
- Sounds difficult, but in practice not too hard

Encyclopaedic context

- These are things that we expect a Hellenized Jew (NT) or an educated Roman (Caesar) to know about
- Sounds difficult, but in practice not too hard
- We can (and must) rely on overt indicators

Encyclopaedic context

- These are things that we expect a Hellenized Jew (NT) or an educated Roman (Caesar) to know about
- Sounds difficult, but in practice not too hard
- We can (and must) rely on overt indicators
- 6.4% of referents

Encyclopedic knowledge

Caesar BG 1.6.3

(7) *Extremum oppidum Allobrogum est proximum=que*
furthest town Allobroges.GEN is closest=and
Helvetiorum finibus Genava
Helvetii.GEN borders.DAT Geneva.NOM
'The furthest town of the Allobroges and the closes to the borders of the Helvetii is Geneva'

Encyclopedic knowledge

Caesar BG 1.10.3

(7) *tres quae circum Aquileiam hiemabant ex*
three who around Aquieia winter.IMPF.3P from
hibernis PRO-SUB educit
winter camp.ABL PRO lead
'He led the three legions that were wintering around Aquileia out of (their) winter camps

No context

- Another straigtforward tag, once it is decided that there is a taggable in the first place
- Not very common (9.8% of NPs)

Complex NPs

Luke 10:16

(8) *ho de eme athetôn athetei ton aposteilanta me*
the but me.ACC refusing refuse.PRS.3SG the sending me.ME
'Whoever refuses me, refuses the one who sent me'

Internal anchoring

- 'Bridging' is often explicitly marked, e.g. by a possessive

Internal anchoring

- 'Bridging' is often explicitly marked, e.g. by a possessive
- In this case too, we want to mark both referents

▸ Example

Embedded discourse referents

- We saw that we can have anaphoric processes inside embeddings that otherwise block anaphoric reference from the outside

Embedded discourse referents

- We saw that we can have anaphoric processes inside embeddings that otherwise block anaphoric reference from the outside
- The reference is not necessarily direct; we find "embedded bridging" as well

Luke 11:33

No one, after lighting a lamp, puts it away in a cellar nor under a basket, but on the lampstand, so that those who enter may see the light.

Our solution

- But we can use a separate set of tags for all DRs outside the main DRS
  - NONSPEC (but QUANT for quantification)
  - NONSPEC_INF
  - NONSPEC_OLD

Our solution

- But we can use a separate set of tags for all DRs outside the main DRS
  - NONSPEC (but QUANT for quantification)
  - NONSPEC_INF
  - NONSPEC_OLD
- No counterparts to ACC-GEN or ACC-SIT as these belong in the main DRS by definition

Table of Contents

What is a regular expression?

- A regular expression (or a regex, or a regexp) is a pattern matching a certain set of strings

What is a regular expression?

- A regular expression (or a regex, or a regexp) is a pattern matching a certain set of strings
- One single regular expression can match many different strings
- This makes them powerful!

What is a regular expression?

- A regular expression (or a regex, or a regexp) is a pattern matching a certain set of strings
- One single regular expression can match many different strings
- This makes them powerful!
- A number of applications support regexes
  - Corpus query engines
  - Text editors
  - ...

Literals

- The regular expression /a/ matches the first occurrence of 'a' in a string
- The same is true for most characters, except a few special ones:
  - \^$. | ? * + and parentheses (normal, curly, square)
- To match special characters you need to "escape" them with a backslash

Practice

- There are lots of online apps what will let you practice regexes
- Let us try http://www.regextester.com/ on this text

[1.] Gallia est omnis divisa in partes tres, quarum unam incolunt Belgae, aliam Aquitani, tertiam qui ipsorum lingua Celtae, nostra Galli appellantur. [2] Hi omnes lingua, institutis, legibus inter se differunt. Gallos ab Aquitanis Garumna flumen, a Belgis Matrona et Sequana dividit. [3] Horum omnium fortissimi sunt Belgae, propterea quod a cultu atque humanitate provinciae longissime absunt, minimeque ad eos mercatores saepe commeant atque ea quae ad effeminandos animos pertinent important, [4] proximique sunt Germanis, qui trans Rhenum incolunt, quibuscum continenter bellum gerunt. Qua de causa Helvetii quoque reliquos Gallos virtute praecedunt, quod fere cotidianis proeliis cum Germanis contendunt, cum aut suis finibus eos prohibent aut ipsi in eorum finibus bellum gerunt.

Sample queries

- Find single character sequences:
  - Find all `a`'s
  - Find all `s`'s
  - Find all opening square brackets
  - Find all commas
  - Find all full stops
- We can sequence characters in the obvious way:
  - Find all `st` sequences
  - Find all stops ending a "paragraph" (i.e. followed by space and a square bracket)

Character classes

- We can define character *classes* that match single characters in a particular class

List enclose the characters in square brackets: `[ae]` matches an a or e

Negated list if you put a caret after the opening square bracket, you will match any character *not* listed: `[^aeiouy` will match any non-vowel

Ranges `[1-9]` matches any digit, `[a-z]` any lower case letter

Wildcard `.` matches any character

Predefined many regex engines have predefined classes

Find ...

- words that end in -es or -is
- paragraph references

Anchors

- It can be useful to match line beginnings and ends

Anchors

- It can be useful to match line beginnings and ends
- This is done with ^and $

Anchors

- It can be useful to match line beginnings and ends
- This is done with ^and $
- How do you find full stops at line ends?

Anchors

- It can be useful to match line beginnings and ends
- This is done with ^and $
- How do you find full stops at line ends?
- We can also match word boundaries with \b

Disjunctions

- If we want to match either of two characters, we can use a character class: `[ae]`

Disjunctions

- If we want to match either of two characters, we can use a character class: `[ae]`
- If we want to match either of two character *strings*, we use a disjuction: `(ha|ah)`

Disjunctions

- If we want to match either of two characters, we can use a character class: `[ae]`
- If we want to match either of two character *strings*, we use a disjuction: `(ha|ah)`
- `([io]s?|u[sm])` will find all 2nd declension forms (and lots of other stuff!)

Disjunctions

- If we want to match either of two characters, we can use a character class: `[ae]`
- If we want to match either of two character *strings*, we use a disjuction: `(ha|ah)`
- `([io]s?|u[sm])` will find all 2nd declension forms (and lots of other stuff!)
- How do we find references to Gauls and Germans in the dative plural?

Quantifiers

- We can attach quantifiers to a character (class):
  - `*` zero or more
  - `?` zero or one
  - `+` one or more
- Less often used
  - `{5}` five times
  - `{3,5}` between three and five times
  - `{3,}` three or more times

Sample regular expressions

- Here are some examples of how regexes can match different forms in a paradigm

Sample regular expressions

- Here are some examples of how regexes can match different forms in a paradigm

  `"amat"`: *amat*

Sample regular expressions

- Here are some examples of how regexes can match different forms in a paradigm

`"amat"`:	*amat*
`"ama[st]"`:	*amas* or *amat*

Sample regular expressions

- Here are some examples of how regexes can match different forms in a paradigm

`"amat"`:	*amat*
`"ama[st]"`:	*amas* or *amat*
`"amat?"`:	*ama* or *amat*

Sample regular expressions

- Here are some examples of how regexes can match different forms in a paradigm

`"amat"`:	*amat*
`"ama[st]"`:	*amas* or *amat*
`"amat?"`:	*ama* or *amat*
`"ama[st]?"`:	*ama*, *amas* or *amat*

Sample regular expressions

- Here are some examples of how regexes can match different forms in a paradigm

`"amat"`:	*amat*
`"ama[st]"`:	*amas* or *amat*
`"amat?"`:	*ama* or *amat*
`"ama[st]?"`:	*ama*, *amas* or *amat*

- But regexes are most useful in dealing with the **morph** and **pos** attributes

More exercises

- Match all inflectional forms of filia!

More exercises

- Match all inflectional forms of filia!
- Try to find all participles and as little else as possible!

More exercises

- Match all inflectional forms of filia!
- Try to find all participles and as little else as possible!

More advanced usage

- Many editors will let you use regexes in powerful find and replace commands

More advanced usage

- Many editors will let you use regexes in powerful find and replace commands
- Let us see how we can extract a text from an xml-file

More advanced usage

- Many editors will let you use regexes in powerful find and replace commands
- Let us see how we can extract a text from an xml-file
- Download a PROIEL xml file and open it in some editor (I use emacs)

More advanced usage

- Many editors will let you use regexes in powerful find and replace commands
- Let us see how we can extract a text from an xml-file
- Download a PROIEL xml file and open it in some editor (I use emacs)
- What happens if I look for `form=".*"`?

More advanced usage

- Many editors will let you use regexes in powerful find and replace commands
- Let us see how we can extract a text from an xml-file
- Download a PROIEL xml file and open it in some editor (I use emacs)
- What happens if I look for `form=".*"`?
- We need `form="[^"]*"`

Captures

- For find and replace commands, we may want to reference (part of) the match in the replace expression

Captures

- For find and replace commands, we may want to reference (part of) the match in the replace expression
- For this we must define *captures* in the match

Captures

- For find and replace commands, we may want to reference (part of) the match in the replace expression
- For this we must define *captures* in the match
- This is often done with (), but in emacs it's \(\)

Captures

- For find and replace commands, we may want to reference (part of) the match in the replace expression
- For this we must define *captures* in the match
- This is often done with (), but in emacs it's \(\)
- We can reference captures from the match in the replace expression with \1, \2, ...
- So we replace `form="\([^"]*\)"` with \1

Table of Contents

Treebanks

- Recall that a treebank is a representation of the syntactic (or semantic) structure of sentences
- A major motivation for syntactic annotation is to be able to query these structures
- Unfortunately, the structures are complex, so the query languages are complex

Availability of data

`http://foni.uio.no/proiel` Browsing, simple single-token querying of all annotations (click “search”)

Availability of data

`http://foni.uio.no/proiel` Browsing, simple single-token querying of all annotations (click "search")

`http://iness.uib.no` Deep querying of morphology and syntax

Availability of data

`http://foni.uio.no/proiel` Browsing, simple single-token querying of all annotations (click "search")

`http://iness.uib.no` Deep querying of morphology and syntax

`http://proiel.github.io/` Versioned source data available for download (under CC-BY-NC-SA) with all annotations and full flexibility

Availability of data

`http://foni.uio.no/proiel` Browsing, simple single-token querying of all annotations (click "search")

`http://iness.uib.no` Deep querying of morphology and syntax

`http://proiel.github.io/` Versioned source data available for download (under CC-BY-NC-SA) with all annotations and full flexibility

- Today's topic is the INESS web page

Availability of data

`http://foni.uio.no/proiel` Browsing, simple single-token querying of all annotations (click "search")

`http://iness.uib.no` Deep querying of morphology and syntax

`http://proiel.github.io/` Versioned source data available for download (under CC-BY-NC-SA) with all annotations and full flexibility

- Today's topic is the INESS web page
- This query language is based on TigerSearch, which is the basis for many other search engines

Availability of data

`http://foni.uio.no/proiel` Browsing, simple single-token querying of all annotations (click "search")

`http://iness.uib.no` Deep querying of morphology and syntax

`http://proiel.github.io/` Versioned source data available for download (under CC-BY-NC-SA) with all annotations and full flexibility

- Today's topic is the INESS web page
- This query language is based on TigerSearch, which is the basis for many other search engines
- But first a few words on the simplest interface

PROIEL queries

- The PROIEL webapp has a small, lightweight query interface

PROIEL queries

- The PROIEL webapp has a small, lightweight query interface
- Useful for finding single words with specific properties
- No support for queries on relations between words

PROIEL queries

- The PROIEL webapp has a small, lightweight query interface
- Useful for finding single words with specific properties
- No support for queries on relations between words
- Nevertheless often useful

Sample PROIEL queries

- all common nouns (in Greek, or in Herodotus)

Sample PROIEL queries

- all common nouns (in Greek, or in Herodotus)
- all subjects

Sample PROIEL queries

- all common nouns (in Greek, or in Herodotus)
- all subjects
- all new information subjects in the Gospels

Sample PROIEL queries

- all common nouns (in Greek, or in Herodotus)
- all subjects
- all new information subjects in the Gospels
- all forms of *conficio*

Sample PROIEL queries

- all common nouns (in Greek, or in Herodotus)
- all subjects
- all new information subjects in the Gospels
- all forms of *conficio*
- all compounds of *facio*

Sample PROIEL queries

- all common nouns (in Greek, or in Herodotus)
- all subjects
- all new information subjects in the Gospels
- all forms of *conficio*
- all compounds of *facio*
- all restrictive relative clauses

Sample PROIEL queries

- all common nouns (in Greek, or in Herodotus)
- all subjects
- all new information subjects in the Gospels
- all forms of *conficio*
- all compounds of *facio*
- all restrictive relative clauses
- all headless relative clauses in subject function

Basics of INESS

- INESS is a portal offering access to many treebanks of different languages
- To use it you need to know about two things
  - The INESS query language (based on TigerSearch)
  - The particulars of the corpus you are looking at, in our case PROIEL
- First, we need to choose the right corpus:
  - Click "Treebank selection"
  - Click e.g. "Latin"
  - You get a list of all available Latin texts
  - By default, you query all but you can toggle this on the left side
- "Documentation" will take you to documentation of the query language

Words and links, nodes and edges

- During annotation, we represented the data as words (aka **nodes**) and links (aka **edges**) between them

Words and links, nodes and edges

- During annotation, we represented the data as words (aka **nodes**) and links (aka **edges**) between them
- We gave nodes **attributes**

  word is the form that occurs in the text

Words and links, nodes and edges

- During annotation, we represented the data as words (aka **nodes**) and links (aka **edges**) between them
- We gave nodes **attributes**

  word is the form that occurs in the text

  lemma is the dictionary entry

Words and links, nodes and edges

- During annotation, we represented the data as words (aka **nodes**) and links (aka **edges**) between them
- We gave nodes **attributes**

word	is the form that occurs in the text
lemma	is the dictionary entry
pos	is the part of speech

Words and links, nodes and edges

- During annotation, we represented the data as words (aka **nodes**) and links (aka **edges**) between them
- We gave nodes **attributes**

word	is the form that occurs in the text
lemma	is the dictionary entry
pos	is the part of speech
morph	is the morphological tag

Words and links, nodes and edges

- During annotation, we represented the data as words (aka **nodes**) and links (aka **edges**) between them
- We gave nodes **attributes**

word	is the form that occurs in the text
lemma	is the dictionary entry
pos	is the part of speech
morph	is the morphological tag

- We used to types of edges, which both got **labels**
  - Primary edges are labelled with syntactic functions such as **sub**, **obj**
  - Secondary edges are labelled with ordinary syntactic functions or special relations

Structure of an attribute query

- Constraints on a single node go inside square brackets and are of the form `atr="val"`, e.g. `[word="sum"]`
- Before the equals sign is an **attribute**, after it comes a **regular expression** enclosed in quotes

Structure of the **pos** attribute

- Recall that **pos** is a string with two characters
  - The first character gives the major part of speech, e.g. pronoun
  - The second gives the subtype, e.g. interrogative, relative etc.
  - "-" in the second field means there is no subtype
- Examples

  pos="Ne" finds all proper nouns

Structure of the **pos** attribute

- Recall that **pos** is a string with two characters
  - The first character gives the major part of speech, e.g. pronoun
  - The second gives the subtype, e.g. interrogative, relative etc.
  - "-" in the second field means there is no subtype
- Examples

  pos="Ne" finds all proper nouns

  pos="Nb" finds all common nouns

Structure of the **pos** attribute

- Recall that **pos** is a string with two characters
  - The first character gives the major part of speech, e.g. pronoun
  - The second gives the subtype, e.g. interrogative, relative etc.
  - "-" in the second field means there is no subtype
- Examples

  pos="Ne" finds all proper nouns

  pos="Nb" finds all common nouns

  pos="N[be]" finds all proper and common nouns

Structure of the **pos** attribute

- Recall that **pos** is a string with two characters
  - The first character gives the major part of speech, e.g. pronoun
  - The second gives the subtype, e.g. interrogative, relative etc.
  - "-" in the second field means there is no subtype
- Examples

pos="Ne"	finds all proper nouns
pos="Nb"	finds all common nouns
pos="N[be]"	finds all proper and common nouns
pos="N."	finds all types of nouns (= in this case, proper and common)

Structure of the **morph** attribute

- **morph** works the same way as **pos** but remember there are ten fields!
- person, number, tense, mood, voice, gender, case, degree, strength, inflection
- this makes for very complicated strings!

Feature values

feature	possible values
person	1, 2, 3
number	s (singular), d (dual), p (plural)
tense	l (pluperfect), a (aorist), p (present), f (future), r (perfect), s (resultative), i (imperfect), t (future perfect), u (past)
mood	m (imperative), n (infintive), o (optative), d (gerund), p (participle), g (gerundive), s (subjunctive), i (indicative), u (supine)
voice	a (active), m (middle), e (middle or passive), p (passive)
gender	m (masculine), f (feminine), n (neuter), o (m. or n.), p (m. or f.), q (m., f. or n.), r (f. or n.)

Feature values

feature	possible values
person	1, 2, 3
number	s (singular), d (dual), p (plural)
tense	l (pluperfect), a (aorist), p (present), f (future), r (perfect), s (resultative), i (imperfect), t (future perfect), u (past)
mood	m (imperative), n (infintive), o (optative), d (gerund), p (participle), g (gerundive), s (subjunctive), i (indicative), u (supine)
voice	a (active), m (middle), e (middle or passive), p (passive)
gender	m (masculine), f (feminine), n (neuter), o (m. or n.), p (m. or f.), q (m., f. or n.), r (f. or n.)
case	n (nominative), a (accusative), g (genitive), d (dative), c (genitive or dative), i (instrumental), b (ablative), l (locative)
degree	p (positive), c (comparative), s (superlative)
strength	w (weak), s (strong), t (weak or strong)
inflection	i (inflecting), n (non-inflecting)

Examples

`"3sria----i"` Third person, singular, perfect, indicative, active, inflecting, e.g. *vidit*

Examples

`"3sria----i"` Third person, singular, perfect, indicative, active, inflecting, e.g. *vidit*

`"---------n"` Non-inflecting, e.g. *ab*

Examples

`"3sria----i"` Third person, singular, perfect, indicative, active, inflecting, e.g. *vidit*

`"---------n"` Non-inflecting, e.g. *ab*

`"-p---mb--i"` Plural, masculine, ablative, inflecting, e.g. *hominibus*

Examples

`"3sria----i"` Third person, singular, perfect, indicative, active, inflecting, e.g. *vidit*

`"---------n"` Non-inflecting, e.g. *ab*

`"-p---mb--i"` Plural, masculine, ablative, inflecting, e.g. *hominibus*

`"-s---qbc-i"` Singular, m./f./n., ablative, comparative, inflecting, e.g. *meliore*

Examples

`"3sria----i"` Third person, singular, perfect, indicative, active, inflecting, e.g. *vidit*

`"---------n"` Non-inflecting, e.g. *ab*

`"-p---mb--i"` Plural, masculine, ablative, inflecting, e.g. *hominibus*

`"-s---qbc-i"` Singular, m./f./n., ablative, comparative, inflecting, e.g. *meliore*

Regular expressions over **morph**

- We are typically not interested in all the fields at once, so . is our friend

Regular expressions over **morph**

- We are typically not interested in all the fields at once, so . is our friend
- For example, let us say we are interested in perfect tense verbs:
  - `[morph="..r.*"]`

Regular expressions over **morph**

- We are typically not interested in all the fields at once, so . is our friend
- For example, let us say we are interested in perfect tense verbs:
  - `[morph="..r.*"]`
- Or superlatives
  - `[morph=".......s.*"]`

Regular expressions over **morph**

- We are typically not interested in all the fields at once, so . is our friend
- For example, let us say we are interested in perfect tense verbs:
  - `[morph="..r.*"]`
- Or superlatives
  - `[morph=".......s.*"]`
- We can combine several constraints. Let us say we are interested in the distribution ablative endings in *-i* or *-e* in third declension adjectives
  - `[morph="......b.*" & pos="A." & word=".*[ie]"]`

Naming nodes

- In some contexts we may want to name the nodes

Naming nodes

- In some contexts we may want to name the nodes
- A node name is an ASCII string preceded by #, e.g. #x

Naming nodes

- In some contexts we may want to name the nodes
- A node name is an ASCII string preceded by #, e.g. #x
- They are attached to the node constraints with a colon
  - `#x:[morph="......b.*" & pos="A." & word=".*[ie]"]`

Naming nodes

- In some contexts we may want to name the nodes
- A node name is an ASCII string preceded by #, e.g. #x
- They are attached to the node constraints with a colon
  - `#x:[morph="......b.*" & pos="A." & word=".*[ie]"]`
- This allows us to give several constraints on a single node
  - `#x:[morph="......b.*"] & #x:[pos="A."] & #x:[word=".*[ie]"]`

Naming nodes

- In some contexts we may want to name the nodes
- A node name is an ASCII string preceded by #, e.g. #x
- They are attached to the node constraints with a colon
  - `#x:[morph=".....b.*" & pos="A." & word=".*[ie]"]`
- This allows us to give several constraints on a single node
  - `#x:[morph=".....b.*"] & #x:[pos="A."] & #x:[word=".*[ie]"]`
- In this case, the two queries are equivalent, but when we deal with syntax, we often **have to** use node names

Tables of results

- A nice effect of named nodes is that they allow us to tabulate the results by running the same query in the “query” view rather than the “sentence” view

Tables of results

- A nice effect of named nodes is that they allow us to tabulate the results by running the same query in the "query" view rather than the "sentence" view
- By default a table is constructed based on all node constraints

Tables of results

- A nice effect of named nodes is that they allow us to tabulate the results by running the same query in the "query" view rather than the "sentence" view
- By default a table is constructed based on all node constraints
- However, if a node name ends with an underscore (_), its constraints are not tabulated

Tables of results

- A nice effect of named nodes is that they allow us to tabulate the results by running the same query in the "query" view rather than the "sentence" view
- By default a table is constructed based on all node constraints
- However, if a node name ends with an underscore (_), its constraints are not tabulated
- So if we wanted to table only the word form and not the morph- or postag:
  - `#x_:[morph="......b.*"] & #x_:[pos="A."] & #x:[word=".*[ie]"] & #x_=#x`

Querying syntax

- The syntactic structure is modelled as a tree, which is a 2-dimensional object and there are two corresponding operators that we need to know about
  - > The vertical dimension, dominance/government

Querying syntax

- The syntactic structure is modelled as a tree, which is a 2-dimensional object and there are two corresponding operators that we need to know about
  - `>` The vertical dimension, dominance/government
  - `.` The horizonal dimension, precedence
- Examples
  - `#x >#y` Node *x* directly dominates node *y*

Querying syntax

- The syntactic structure is modelled as a tree, which is a 2-dimensional object and there are two corresponding operators that we need to know about
  - `>` The vertical dimension, dominance/government
  - `.` The horizonal dimension, precedence
- Examples
  - `#x >#y` Node *x* directly dominates node *y*
  - `#x . #y` Node *x* directly precedes node *y*
- * gives us the generalization ('transitive closure') of these operators
  - `#x >* #y` Node *x* dominates node *y*

Querying syntax

- The syntactic structure is modelled as a tree, which is a 2-dimensional object and there are two corresponding operators that we need to know about
  - `>` The vertical dimension, dominance/government
  - `.` The horizonal dimension, precedence
- Examples
  - `#x >#y` Node *x* directly dominates node *y*
  - `#x . #y` Node *x* directly precedes node *y*
- * gives us the generalization ('transitive closure') of these operators
  - `#x >* #y` Node *x* dominates node *y*
  - `#x .* #y` Node *x* precedes node *y*

Labelled dominance

- In most cases we are not interested in pure dominance but in specific syntactic **functions**, which are captured in **labels**

Labelled dominance

- In most cases we are not interested in pure dominance but in specific syntactic **functions**, which are captured in **labels**
- To use them properly you need to carefully study the syntactic guidelines of the corpus you are using

Deictics inside complement clauses

- When e.g. *nunc* occurs within reported speech there is ambiguity whether it refers to the speaker's now or the narrator's?

Deictics inside complement clauses

- When e.g. *nunc* occurs within reported speech there is ambiguity whether it refers to the speaker's now or the narrator's?

Embedded *nunc*

#x >comp #c & #c >* [word="nunc"]

- The first part ensures that #c has the COMP function

Deictics inside complement clauses

- When e.g. *nunc* occurs within reported speech there is ambiguity whether it refers to the speaker's now or the narrator's?

Embedded *nunc*

#x >comp #c & #c >* [word="nunc"]

- The first part ensures that #c has the COMP function
- The second part ensures that *nunc* is dominated by #c

Deictics inside complement clauses

- When e.g. *nunc* occurs within reported speech there is ambiguity whether it refers to the speaker's now or the narrator's?

Embedded *nunc*

#x >comp #c & #c >* [word="nunc"]

- The first part ensures that #c has the COMP function
- The second part ensures that *nunc* is dominated by #c

Predicate-subject pairs

#x >sub #y finds all subjects

Predicate-subject pairs

`#x >sub #y` finds all subjects

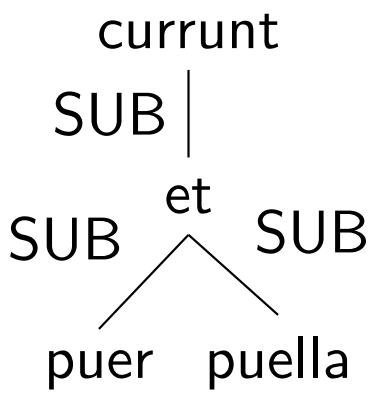
```
#x:[lemma=".*"] >sub #y:[lemma=".*"]
```
tabulates per lemma

Predicate-subject pairs

`#x >sub #y` finds all subjects

```
#x:[lemma=".*"] >sub #y:[lemma=".*"]
```
tabulates per lemma

- False positives because of the analysis of coordination

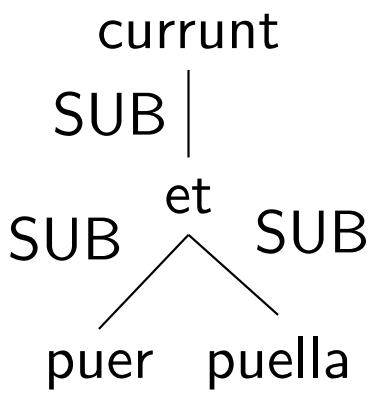


Predicate-subject pairs

`#x >sub #y` finds all subjects

`#x:[lemma=".*"] >sub #y:[lemma=".*"]` tabulates per lemma

- False positives because of the analysis of coordination



- Fixed with `#x:[lemma=".*"] >((sub)+) #y:[lemma=".*"] & #x:[pos="V-"] & #y:[pos!="C-"]`

A preliminary: studying word order without corpora

	Luke + Acts		Luke only	
	Rife(1933)	Davison(1989)	Rife(1933)	Kirk(2012)
VSO	15	20	9	14
SVO	50	56	19	13
SOV	9	8	8	5
VOS	3	4	2	3
OVS	6	6	1	1
OSV	1	1	0	1

A preliminary: studying word order without corpora

	Luke + Acts		Luke only	
	Rife(1933)	Davison(1989)	Rife(1933)	Kirk(2012)
VSO	15	20	9	14
SVO	50	56	19	13
SOV	9	8	8	5
VOS	3	4	2	3
OVS	6	6	1	1
OSV	1	1	0	1

- All the authors claim to report the same basic fact: word order in declarative main clauses where there is an NP subject and object

A preliminary: studying word order without corpora

	Luke + Acts		Luke only	
	Rife(1933)	Davison(1989)	Rife(1933)	Kirk(2012)
VSO	15	20	9	14
SVO	50	56	19	13
SOV	9	8	8	5
VOS	3	4	2	3
OVS	6	6	1	1
OSV	1	1	0	1

- All the authors claim to report the same basic fact: word order in declarative main clauses where there is an NP subject and object
- How can we agree on higher level analyses if the facts are so unclear?

What went wrong?

- Rife (1933): "The investigation was limited to main declarative clauses where both subject and object are substantives."

What went wrong?

- Rife (1933): "The investigation was limited to main declarative clauses where both subject and object are substantives."
  - Which text?

What went wrong?

- Rife (1933): "The investigation was limited to main declarative clauses where both subject and object are substantives."
  - Which text?
  - How are terms defined?

What went wrong?

- Rife (1933): "The investigation was limited to main declarative clauses where both subject and object are substantives."
  - Which text?
  - How are terms defined?
- Davison (1989): "clauses ...which contained at least one nominative noun, one accusative noun and one indicative verb ...Verbs normally followed by a genitive or a dative were traced using a concordance"
  - Same problem, although the text is at least available electronically

Being specific about it

- The criteria of Kirk (2012)

Being specific about it

- The criteria of Kirk (2012)
  - The clause contains at least an S(ubject), V(erb) and O(bject)

Being specific about it

- The criteria of Kirk (2012)
  - The clause contains at least an S(ubject), V(erb) and O(bject)
  - The clause is continuous

Being specific about it

- The criteria of Kirk (2012)
  - The clause contains at least an S(ubject), V(erb) and O(bject)
  - The clause is continuous
  - S and O are not embedded in a participial clause

Being specific about it

- The criteria of Kirk (2012)
  - The clause contains at least an S(ubject), V(erb) and O(bject)
  - The clause is continuous
  - S and O are not embedded in a participial clause
  - The verb assigns accusative, genitive, or dative to an argument that is a patient or theme

Being specific about it

- The criteria of Kirk (2012)
  - The clause contains at least an S(ubject), V(erb) and O(bject)
  - The clause is continuous
  - S and O are not embedded in a participial clause
  - The verb assigns accusative, genitive, or dative to an argument that is a patient or theme
  - The V consists of one word (no periphrastic forms, modal embeddings or light verbs)

Being specific about it

- The criteria of Kirk (2012)
  - The clause contains at least an S(ubject), V(erb) and O(bject)
  - The clause is continuous
  - S and O are not embedded in a participial clause
  - The verb assigns accusative, genitive, or dative to an argument that is a patient or theme
  - The V consists of one word (no periphrastic forms, modal embeddings or light verbs)
  - S and O are determiner phrases (this includes nominalizations) or quantifier phrases, and not clausal

Being specific about it

- The criteria of Kirk (2012)
  - The clause contains at least an S(ubject), V(erb) and O(bject)
  - The clause is continuous
  - S and O are not embedded in a participial clause
  - The verb assigns accusative, genitive, or dative to an argument that is a patient or theme
  - The V consists of one word (no periphrastic forms, modal embeddings or light verbs)
  - S and O are determiner phrases (this includes nominalizations) or quantifier phrases, and not clausal
  - S and O are continuous strings

Being specific about it

- The criteria of Kirk (2012)
  - The clause contains at least an S(ubject), V(erb) and O(bject)
  - The clause is continuous
  - S and O are not embedded in a participial clause
  - The verb assigns accusative, genitive, or dative to an argument that is a patient or theme
  - The V consists of one word (no periphrastic forms, modal embeddings or light verbs)
  - S and O are determiner phrases (this includes nominalizations) or quantifier phrases, and not clausal
  - S and O are continuous strings
- Admirably explicit, can be turned into INESS queries!

Being specific about it

- The criteria of Kirk (2012)
  - The clause contains at least an S(ubject), V(erb) and O(bject)
  - The clause is continuous
  - S and O are not embedded in a participial clause
  - The verb assigns accusative, genitive, or dative to an argument that is a patient or theme
  - The V consists of one word (no periphrastic forms, modal embeddings or light verbs)
  - S and O are determiner phrases (this includes nominalizations) or quantifier phrases, and not clausal
  - S and O are continuous strings
- Admirably explicit, can be turned into INESS queries!
- This way we can do the query, look at the results and possibly modify it

Word order in main clauses - Kirk's criteria

- Finding a main clause
  - Unbroken sequence of PRED relations: `#r >((pred)*) #v`

Word order in main clauses - Kirk's criteria

- Finding a main clause
  - Unbroken sequence of PRED relations: `#r >((pred)*) #v`
  - Under the root: `!(#x > #r)`

Word order in main clauses - Kirk's criteria

- Finding a main clause
  - Unbroken sequence of PRED relations: `#r >((pred)*) #v`
  - Under the root: `!(#x > #r)`
  - Not an elided verb: `#v:[pos="V-"]`
- With a subject and an object: `#v >sub #s & #v >((obj)|(obl)) #o`

Word order in main clauses - Kirk's criteria

- Finding a main clause
  - Unbroken sequence of PRED relations: `#r >((pred)*) #v`
  - Under the root: `!(#x > #r)`
  - Not an elided verb: `#v:[pos="V-"]`
- With a subject and an object: `#v >sub #s & #v >((obj)|(obl)) #o`
- ...which are nominals : `#s:[pos="[AN]."] & #o:[pos="[AN]."]`

Word order in main clauses - Kirk's criteria

- Finding a main clause
  - Unbroken sequence of PRED relations: `#r >((pred)*) #v`
  - Under the root: `!(#x > #r)`
  - Not an elided verb: `#v:[pos="V-"]`
- With a subject and an object: `#v >sub #s & #v >((obj)|(obl)) #o`
- ...which are nominals : `#s:[pos="[AN]."] & #o:[pos="[AN]."]`
- ...and not discontinuous: should have been `_cont(#s) & _cont(#o)` (does not currently work!)

Word order in main clauses - Kirk's criteria

- Finding a main clause
  - Unbroken sequence of PRED relations: `#r >((pred)*) #v`
  - Under the root: `!(#x > #r)`
  - Not an elided verb: `#v:[pos="V-"]`
- With a subject and an object: `#v >sub #s & #v >((obj)|(obl)) #o`
- ...which are nominals : `#s:[pos="[AN]."] & #o:[pos="[AN]."]`
- ...and not discontinuous: should have been `_cont(#s) & _cont(#o)` (does not currently work!)
- No auxiliares: `!(#v >aux #aux:[pos=".+"] )`

Word order in main clauses - Kirk's criteria

- Finding a main clause
  - Unbroken sequence of PRED relations: `#r >((pred)*) #v`
  - Under the root: `!(#x > #r)`
  - Not an elided verb: `#v:[pos="V-"]`
- With a subject and an object: `#v >sub #s & #v >((obj)|(obl)) #o`
- ...which are nominals : `#s:[pos="[AN]."] & #o:[pos="[AN]."]`
- ...and not discontinuous: should have been `_cont(#s) & _cont(#o)` (does not currently work!)
- No auxiliares: `!(#v >aux #aux:[pos=".+"] )`
- Build your query step by step!

The result

```
#r >((pred)*) #v & !(#x > #r) & #v:[pos="V-"] & #v >sub #s
& #v >((obj)|(obl)) #o & #s:[pos="[AN]."] &
#o:[pos="[AN]."] & !(#v >aux #aux:[pos=".+"] )
```

The result

```
#r >((pred)*) #v & !(#x > #r) & #v:[pos="V-"] & #v >sub #s
& #v >((obj)|(obl)) #o & #s:[pos="[AN]."] &
#o:[pos="[AN]."] & !(#v >aux #aux:[pos=".+"] )
```

- Pretty daunting, but after all we found the data material for a whole dissertation in a matter of minutes

The result

```
#r >((pred)*) #v & !(#x > #r) & #v:[pos="V-"] & #v >sub #s
& #v >((obj)|(obl)) #o & #s:[pos="[AN]."] &
#o:[pos="[AN]."] & !(#v >aux #aux:[pos=".+"] )
```

- Pretty daunting, but after all we found the data material for a whole dissertation in a matter of minutes
- Moreover, it is flexible - we can tweak the query and look at various other definitions of the source material

The result

```
#r >((pred)*) #v & !(#x > #r) & #v:[pos="V-"] & #v >sub #s
& #v >((obj)|(obl)) #o & #s:[pos="[AN]."] &
#o:[pos="[AN]."] & !(#v >aux #aux:[pos=".+"] )
```

- Pretty daunting, but after all we found the data material for a whole dissertation in a matter of minutes
- Moreover, it is flexible - we can tweak the query and look at various other definitions of the source material
- That is for the exercises!

Exercises

- Change the query so that we allow pronominal subjects and objects

Exercises

- Change the query so that we allow pronominal subjects and objects
- Change the query so that we only look at complement clauses

Exercises

- Change the query so that we allow pronominal subjects and objects
- Change the query so that we only look at complement clauses
- Change the query so that we only look at adverbial clauses

Exercises

- Change the query so that we allow pronominal subjects and objects
- Change the query so that we only look at complement clauses
- Change the query so that we only look at adverbial clauses
- There are six possible permutations of S, V and O. Express these permutations as INESS queries that can be added to our base query

Exercises

- Change the query so that we allow pronominal subjects and objects
- Change the query so that we only look at complement clauses
- Change the query so that we only look at adverbial clauses
- There are six possible permutations of S, V and O. Express these permutations as INESS queries that can be added to our base query
- Change the query so that you find orders of subject, verb and complement clause (instead of object) and specify different types of complement clauses
  - accusative with infinitive
  - finite complement clause