Computational Syntax, Dependency Parsing

Koldo Gojenola. HAP/LAP.

Bibliography

- Natural Language Processing (forthcoming, MIT Press). Jacob Eisenstein. https://github.com/jacobeisenstein/gt-nlp-class/blob/master/notes/eisenstein-nlp-notes.pdf
- Natural Language Processing with Python. NLTK Book. Chapter 8. Analyzing Sentence Structure. http://www.nltk.org/book/ch08.html
- Dependency Parsing (Synthesis Lectures on Human Language Technologies), 2009 by Sandra Kubler, Ryan McDonald, Joakim Nivre. Morgan & Claypool Publishers

Index

1. Dependency Parsing
   - Representation
   - Dependency analyzers
   - Universal Dependencies (UD)

Dependency Parsing

- Representation

Dependency Parsing

- Representation
- Analyzers
  - Knowledge-based
  - Data-driven (statistical methods)

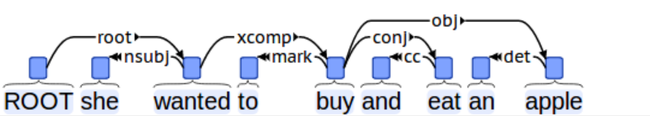
Dependency Parsing

- Representation
- Analyzers
  - Knowledge-based
  - Data-driven (statistical methods)
- Treebanks

Dependency Parsing

- Representation
- Analyzers
  - Knowledge-based
  - Data-driven (statistical methods)
- Treebanks
- Publicly available dependency analyzers

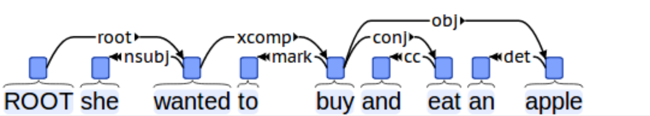
Representation



- An arc from i (head) to j (dependent)
- Special arc to the head of the sentence: **root**

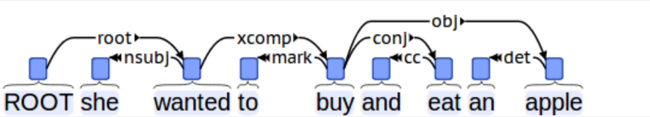
Projectivity

- Projective tree: no crossing links



Projectivity

- Projective tree: no crossing links



- Non-projective trees:
  Lucia ate a pizza yesterday which was vegetarian

Analyzers: data-driven

Procedure:

- A *treebank* for training
- Obtain a model using machine learning
- Apply the model to new sentences

Analyzers: data-driven

Procedure:

- A *treebank* for training
- Obtain a model using machine learning
- Apply the model to new sentences

Parsing:

Analyzers: data-driven

Procedure:

- A *treebank* for training
- Obtain a model using machine learning
- Apply the model to new sentences

Parsing:

- Graph-based methods
- Transition-based methods
- Combinations: hybrid systems
- Algorithms:
  - Chu-Liu-Edmonds algorithm (projective and non-projective trees)
  - CKY and Eisner's algorithms (projective trees)

Graph-based dependency analyzers

Basic idea:

- A sentence is represented as a graph

Graph-based dependency analyzers

Basic idea:

- A sentence is represented as a graph
- Learning: how to calculate scores for arcs

Graph-based dependency analyzers

Basic idea:

- A sentence is represented as a graph
- Learning: how to calculate scores for arcs
- Parsing: find the graph (dependency tree) with the best score

Graph-based dependency analyzers

Basic idea:

- A sentence is represented as a graph
- Learning: how to calculate scores for arcs
- Parsing: find the graph (dependency tree) with the best score

Features:

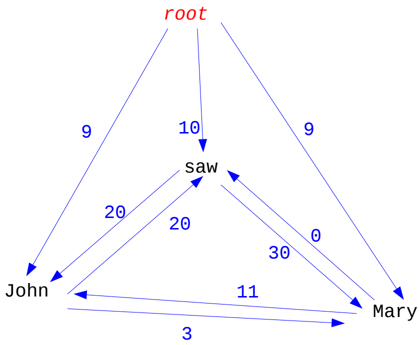
- Global learning
- Exhaustive search (all options)

Graph-based dependency analyzers

Sentence: **John saw Mary**

Graph-based dependency analyzers

Sentence: **John saw Mary**



MST: Maximum Spanning Tree

Main ideas:

- Score for a tree: sum over all arcs (first order, arc factored)

MST: Maximum Spanning Tree

Main ideas:

- Score for a tree: sum over all arcs (first order, arc factored)
- Find the tree with maximum score (containing all words)

MST: Maximum Spanning Tree

Main ideas:

- Score for a tree: sum over all arcs (first order, arc factored)
- Find the tree with maximum score (containing all words)
- Use machine learning to calculate the arc weights (e.g. MIRA algorithm)

Categories (POS)

Second order, third order dependency parsing (Eisenstein 2019)

First order	$h \quad m$	
Second order	$h \quad s \quad m$	$g \quad h \quad m$
Third order	$g \quad h \quad s \quad m$	$h \quad t \quad s \quad m$

Figure 11.6: Feature templates for higher-order dependency parsing

Transition-based dependency analyzers

Main idea:

- Define a transition-based system, to construct a dependency tree from a sentence

Transition-based dependency analyzers

Main idea:

- Define a transition-based system, to construct a dependency tree from a sentence
- Learning: knowing a history of transitions, decide which one is the next transition

Transition-based dependency analyzers

Main idea:

- Define a transition-based system, to construct a dependency tree from a sentence
- Learning: knowing a history of transitions, decide which one is the next transition
- Analysis: given a model, construct the best transition-sequence

Transition-based dependency analyzers

Main idea:

- Define a transition-based system, to construct a dependency tree from a sentence
- Learning: knowing a history of transitions, decide which one is the next transition
- Analysis: given a model, construct the best transition-sequence

Main features:

- Local learning
- *Greedy* search

Transition-based dependency analyzers: shift-reduce deterministic algorithm

Two data-structures:

- A stack, with analyzed elements

Transition-based dependency analyzers: shift-reduce deterministic algorithm

Two data-structures:

- A stack, with analyzed elements
- Sequence of remaining input words

Transition-based dependency analyzers: shift-reduce deterministic algorithm

Two data-structures:

- A stack, with analyzed elements
- Sequence of remaining input words

4 transitions:

- Shift

Transition-based dependency analyzers: shift-reduce deterministic algorithm

Two data-structures:

- A stack, with analyzed elements
- Sequence of remaining input words

4 transitions:

- Shift
- Left-arc (dependency)

Transition-based dependency analyzers: shift-reduce deterministic algorithm

Two data-structures:

- A stack, with analyzed elements
- Sequence of remaining input words

4 transitions:

- Shift
- Left-arc (dependency)
- Right-arc (dependency)
- Reduce

Transition-based dependency analyzers: arc standard transition system

3 transitions:

- Shift: move the first item from the input buffer on to the top of the stack

Transition-based dependency analyzers: arc standard transition system

3 transitions:

- Shift: move the first item from the input buffer on to the top of the stack
- ARC-LEFT: create a new left-facing arc of type r between the item on the top of the stack and the first item in the input buffer. The head of this arc is j, which remains at the front of the input buffer. The arc $j \rightarrow (r)\ i$ is added to A

Transition-based dependency analyzers: arc standard transition system

3 transitions:

- Shift: move the first item from the input buffer on to the top of the stack
- ARC-LEFT: create a new left-facing arc of type r between the item on the top of the stack and the first item in the input buffer. The head of this arc is j, which remains at the front of the input buffer. The arc $j \rightarrow (r)\ i$ is added to A
- ARC-RIGHT: creates a new right-facing arc of type r between the item on the top of the stack and the first item in the input buffer. The head of this arc is i, which is "popped" from the stack and pushed to the front of the input buffer. The arc $j \rightarrow (r)\ i$ is added to A

Arc-standard transition system

Example of arc-standard dependency parsing (Eisenstein 2019)

	σ	β	action	arc added to $\mathcal{A}$
1.	[ROOT]	*they like bagels with lox*	SHIFT	
2.	[ROOT, *they*]	*like bagels with lox*	ARC-LEFT	(*they* $\leftarrow$ *like*)
3.	[ROOT]	*like bagels with lox*	SHIFT	
4.	[ROOT, *like*]	*bagels with lox*	SHIFT	
5.	[ROOT, *like*, *bagels*]	*with lox*	SHIFT	
6.	[ROOT, *like*, *bagels*, *with*]	*lox*	ARC-LEFT	(*with* $\leftarrow$ *lox*)
7.	[ROOT, *like*, *bagels*]	*lox*	ARC-RIGHT	(*bagels* $\rightarrow$ *lox*)
8.	[ROOT, *like*]	*bagels*	ARC-RIGHT	(*like* $\rightarrow$ *bagels*)
9.	[ROOT]	*like*	ARC-RIGHT	(ROOT $\rightarrow$ *like*)
10.	[ROOT]	$\varnothing$	DONE	

Table 11.2: Arc-standard derivation of the unlabeled dependency parse for the input *they like bagels with lox*.

Transition-based dependency analyzers: arc eager transition system

4 transitions:

- Shift: as before

Transition-based dependency analyzers: arc eager transition system

4 transitions:

- Shift: as before
- ARC-LEFT: as before

Transition-based dependency analyzers: arc eager transition system

4 transitions:

- Shift: as before
- ARC-LEFT: as before
- ARC-RIGHT: right dependents can be attached before all of their dependents have been found. Rather than removing the modifier from both the buffer and stack, the ARC-RIGHT action pushes the modifier on to the stack, on top of the head.
- A new REDUCE action is introduced, which can remove elements from the stack if they already have a parent in A

Arc-standard transition system

Example of arc-standard dependency parsing (Eisenstein 2019)

	σ	β	action	arc added to $\mathcal{A}$
1.	[ROOT]	*they like bagels with lox*	SHIFT	
2.	[ROOT, *they*]	*like bagels with lox*	ARC-LEFT	(*they* $\leftarrow$ *like*)
3.	[ROOT]	*like bagels with lox*	ARC-RIGHT	(ROOT $\rightarrow$ *like*)
4.	[ROOT, *like*]	*bagels with lox*	ARC-RIGHT	(*like* $\rightarrow$ *bagels*)
5.	[ROOT, *like, bagels*]	*with lox*	SHIFT	
6.	[ROOT, *like, bagels, with*]	*lox*	ARC-LEFT	(*with* $\leftarrow$ *lox*)
7.	[ROOT, *like, bagels*]	*lox*	ARC-RIGHT	(*bagels* $\rightarrow$ *lox*)
8.	[ROOT, *like, bagels, lox*]	$\varnothing$	REDUCE	
9.	[ROOT, *like, bagels*]	$\varnothing$	REDUCE	
10.	[ROOT, *like*]	$\varnothing$	REDUCE	
11.	[ROOT]	$\varnothing$	DONE	

Table 11.3: Arc-eager derivation of the unlabeled dependency parse for the input *they like bagels with lox*.

Treebanks

- Constituency-based:

Treebanks

- Constituency-based:
  - Penn Treebank (English)
  - Bulgarian: BulTreebank
  - Chinese: Penn Chinese Treebank, Sinica Treebank
  - German: TIGER/NEGRA, TuBa-D/Z
  - Spanish: Cast3LB
  - and many more

Treebanks

- Constituency-based:
  - Penn Treebank (English)
  - Bulgarian: BulTreebank
  - Chinese: Penn Chinese Treebank, Sinica Treebank
  - German: TIGER/NEGRA, TuBa-D/Z
  - Spanish: Cast3LB
  - and many more
- Dependency-based

Treebanks

- Constituency-based:
  - Penn Treebank (English)
  - Bulgarian: BulTreebank
  - Chinese: Penn Chinese Treebank, Sinica Treebank
  - German: TIGER/NEGRA, TuBa-D/Z
  - Spanish: Cast3LB
  - and many more
- Dependency-based
  - Arabic: Prague Arabic Dependency Treebank
  - Czech: Prague Dependency Treebank
  - Danish: Danish Dependency Treebank
  - Portuguese: Bosque: Floresta sinta(c)tica
  - Slovene: Slovene Dependency Treebank
  - Turkish: METU-Sabanci Turkish Treebank

Dependency analyzers:

- Graph-based: MST (Ryan McDonald et al.)

Dependency analyzers:

- Graph-based: MST (Ryan McDonald et al.)
- Transition-based: MaltParser (Joakim Nivre et al.)

Dependency analyzers:

- Graph-based: MST (Ryan McDonald et al.)
- Transition-based: MaltParser (Joakim Nivre et al.)
- Hybrid systems: Mate (Bernd Bohnet) and ZPar (Yue Zhang)
- And many more (neural parsers, ...)

Annotated example

text = These dogs like children.

ID	wordform	lemma	POS	features	head	deprel	offset
1	*These*	*these*	*DET*	*Num = Plur|PronType = Dem*	2	*det*	*Range* = 0 : 5
2	*dogs*	*dog*	*NOUN*	*Num = Plur*	4	*nsubj*	*Range* = 6 : 10
3	*eat*	*eat*	*VERB*	*M = Ind|T = Pres|VForm = Fin*	0	*root*	*Range* = 11 : 14
4	*meat*	*meat*	*NOUN*	*Num = Sing*	3	*obj*	*Range* = 15 : 19
5	.	.	*PUNCT*	_	3	*punct*	*Range* = 19 : 20

Universal Dependencies (UD). Introduction

Universal Dependencies (UD). Introduction

- Universal Dependencies: standard model to annotate treebanks of different languages

Universal Dependencies (UD). Introduction

- Universal Dependencies: standard model to annotate treebanks of different languages
- From 2008, there have been several steps (Stanford Dependencies, Clear, Google UD, Stanford UD, Hamlet).

Universal Dependencies (UD). Introduction

- Universal Dependencies: standard model to annotate treebanks of different languages
- From 2008, there have been several steps (Stanford Dependencies, Clear, Google UD, Stanford UD, Hamlet).
- EACL-2014: start of the project.

Universal Dependencies (UD). Introduction

- Universal Dependencies: standard model to annotate treebanks of different languages
- From 2008, there have been several steps (Stanford Dependencies, Clear, Google UD, Stanford UD, Hamlet).
- EACL-2014: start of the project.
- Members: Joakim Nivre, Dan Zeman, Christopher Manning, Filip Ginter, ...
- Shared tasks: 2017 and 2018
- Guidelines: http://universaldependencies.org/guidelines.html

Languages

- 100 languages

  Amharic, Ancient Greek, Arabic, Armenian, Basque, Bulgarian, Catalan, Chinese, Croatian, Czech, Danish, Dutch, English, Estonian, Finnish, French, German, Greek, Hebrew, Hindi, Hungarian, Indonesian, Irish, Italian, Japanese, Kazakh, Korean, Latin, Norwegian, Persian, Polish, Portuguese, Romanian, Slovenian, Spanish, Swedish, Tamil, Turkish.

- Some of them were directly annotated based on UDs, some others were semiautomatically converted from already existing treebanks

Main ideas

- Do not annotate the same thing in different ways
- Do not make different things look the same
- Heads are semantic elements: (content words).

3 levels of annotation

- Tokenization
  Lexicalist view: Words = elements separated by whitespace.
  Multiword expressions are separated into words
- Morphology
  Categories and morphological features are annotated using Interset as a model
- Syntax
  Heads: semantic elements (content words)

Categories (POS)

Universal POS tags

These tags mark the core part-of-speech categories. To distinguish additional lexical and grammatical properties of words, use the universal features.

Open class words	Closed class words	Other
ADJ	ADP	PUNCT
ADV	AUX	SYM
INTJ	CONJ	X
NOUN	DET	
PROPN	NUM	
VERB	PART	
	PRON	
	SCONJ	

Alphabetical listing

- ADJ: adjective
- ADP: adposition
- ADV: adverb
- AUX: auxiliary verb
- CONJ: coordinating conjunction
- DET: determiner
- INTJ: interjection
- NOUN: noun
- NUM: numeral
- PART: particle
- PRON: pronoun
- PROPN: proper noun
- PUNCT: punctuation
- SCONJ: subordinating conjunction
- SYM: symbol
- VERB: verb
- X: other

Features

Universal features

For core part-of-speech categories, see the universal POS tags. The features listed here distinguish additional lexical and grammatical properties of words, not covered by the POS tags.

Lexical features	Inflectional features	
	*Nominal**	*Verbal**
PronType	Gender	VerbForm
NumType	Animacy	Mood
Poss	NounClass	Tense
Reflex	Number	Aspect
Foreign	Case	Voice
Abbr	Definite	Evident
	Degree	Polarity
		Person
		Polite
		Clusivity

Index: A abbreviation, abessive, ablative, absolute superlative, absolutive, accusative, active, additive, adessive, admirative, adverbial participle, affirmative, allative, animate, antipassive, aoris article, aspect, associative, **B** bantu noun class, benefactive, **C** cardinal, case, causative case, causative voice, clusivity, collective noun, collective numeral, collective pronominal, comitative, common gender, comparative case, comparative degree, complex definiteness, conditional, conjunctive, considerative, construct state, converb, count plural, counting form, **D** dative, definite, definiteness, degree of comparison, delative, demonstrative, desiderative, destinative, direct case, direct voice, directional allative, distributive case, distributive numeral, dual, **E** elative, elevated referent, emphatic, equative case, equative degree, ergative, essive, evidentiality, exclamative, exclusive, **F** factive, feminine, finite verb, first person, firsthand, foreign word, formal, fourth persor fraction, frequentative, future, **G** gender, genitive, gerund, gerundive, greater paucal, greater plura **H** habitual, human, humbled speaker, **I** illative, imperative, imperfect tense, imperfective aspect, inanimate, inclusive, indefinite, indefinite pronominal, indicative, inessive, infinitive, informal, injunctive, instructive, instrumental, interrogative, inverse number, inverse voice, iterative, **J** jussiv **L** lative, locative, **M** masculine, masdar, mass noun, middle voice, modality, mood, motivative, multiplicative numeral, **N** narrative, necessitative, negative polarity, negative pronominal, neuter, nominative, non-finite verb, non-firsthand, non-human, non-specific indefinite, noun class, numbe numeral type, **O** oblique case, optative, ordinal, **P** participle, partitive, passive, past, past perfect, paucal, perfective aspect, perlative, person, personal, pluperfect, plural, plurale tantum, polarity, politeness, positive degree, positive polarity, possessive, potential, present, preterite, progressive, prolative, pronominal type, prospective, purposive case, purposive mood, **Q** quantifier, quantitativ plural, quotative, **R** range numeral, reciprocal pronominal, reciprocal voice, reduced definiteness, reflexive, register, relative, **S** second person, set numeral, singular, singulare tantum, specific indefinite, subjunctive, sublative, superessive, superlative, supine, **T** temporal, tense, terminal

Dependencies

The upper part of the table follows the main organizing principles of the UD taxonomy such that *rows* correspond to functional categ relation to the head (core arguments of clausal predicates, non-core dependents of clausal predicates, and dependents of nominals) *columns* correspond to structural categories of the dependent (nominals, clauses, modifier words, function words). The lower part of lists relations that are not dependency relations in the narrow sense.

	Nominals	Clauses	Modifier words	Function Words
Core arguments	nsubj obj iobj	csubj ccomp xcomp		
Non-core dependents	obl vocative expl dislocated	advcl	advmod* discourse	aux cop mark
Nominal dependents	nmod appos nummod	acl	amod	det clf case
Coordination	**MWE**	**Loose**	**Special**	**Other**
conj cc	fixed flat compound	list parataxis	orphan goeswith reparandum	punct root dep

Dependencies

Alphabetical listing

- acl: clausal modifier of noun (adjectival clause)
- advcl: adverbial clause modifier
- advmod: adverbial modifier
- amod: adjectival modifier
- appos: appositional modifier
- aux: auxiliary
- case: case marking
- cc: coordinating conjunction
- ccomp: clausal complement
- clf: classifier
- compound: compound
- conj: conjunct
- cop: copula
- csubj: clausal subject
- dep: unspecified dependency
- det: determiner
- discourse: discourse element
- dislocated: dislocated elements
- expl: expletive
- fixed: fixed multiword expression
- flat: flat multiword expression
- goeswith: goes with
- iobj: indirect object
- list: list
- mark: marker
- nmod: nominal modifier
- nsubj: nominal subject
- nummod: numeric modifier
- obj: object
- obl: oblique nominal
- orphan: orphan
- parataxis: parataxis
- punct: punctuation
- reparandum: overridden disfluency
- root: root

Categories (POS)

Example of annotation (I) (Eisenstein 2019)

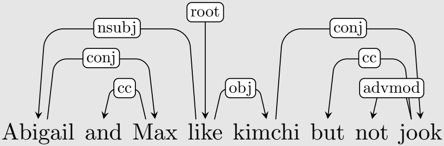


Figure 11.2: In the Universal Dependencies annotation system, the left-most item of a coordination is the head.

Categories (POS)

Example of annotation (II) (Eisenstein 2019)

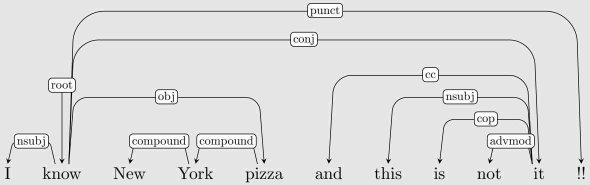


Figure 11.3: A labeled dependency parse from the English UD Treebank (reviews-3613-0006)

https://universaldependencies.github.io/docs/

UD Treebanks

Amharic	-	-	-	-
Ancient Greek	244K			
Ancient Greek-PROIEL	206K	-		
Arabic	282K	-		
Basque	121K			
Bulgarian	156K			
Catalan	-		-	-
Croatian	87K	-		
Czech	1,503K			
Danish	100K			
Dutch	200K	-		
English	254K			
English-ESL	-	-	?	-
Estonian	9K	-		
Finnish	181K			
Finnish-FTB	159K	-		
French	389K			
Galician	-			?
German	293K	-		
Gothic	56K	-		
Greek	59K			
Hebrew	115K	-		
Hindi	351K	-		
Hungarian	26K			
Indonesian	121K	-		
Irish	23K			
Italian	252K			
Japanese-KTC	267K			
Kazakh	-		-	-
Korean	-	-	-	-
Latin	47K	-		
Latin-ITT	259K	-		
Latin-PROIEL	165K	-		
Norwegian	311K			
Old Church Slavonic	57K	-		
Persian	151K			
Polish	83K	-		
Portuguese	212K	-		
Romanian	12K			
Slovenian	140K			
Spanish	423K			
Swedish	96K			
Tamil	9K	-		
Turkish	-		-	-
Ukrainian	-	-		-

Lots of tools for working with UD

- UDAPI
  - Libraries for various UD and CoNLL-U-related operations in several programming languages
    Java, Perl, Python
  - License: GPL, Perl
  - Homepage: http://udapi.github.io/
- UDPipe
  - Trainable pipeline for tokenization, tagging, lemmatization and parsing of CoNLL-U files
  - License: MPL 2.0 (open source)
  - Homepage: Homepage: http://ufal.mff.cuni.cz/udpipe
    Example: echo “Podemos suspendió a los ediles afines a Carmena por temor a un “efecto contagio”” | ./udpipe −−tokenize −−tag −−parse ../../models/udpipe-ud-2.0-170801/spanish-ancora-ud-2.0-170801.udpipe
  - On-line service: http://lindat.mff.cuni.cz/services/udpipe/
    Example: ‘Alemania y Francia comienzan a exigir protecciones médicas en lugares cerrados y Feijóo lo sugiere en España. La escasez de producción y el precio son los principales obstáculos.
- Lots of works to do, both from linguistics and informatics

Tools for inspection, modification of dependency trees

- Lots of tools available for parsing and for dealing with dependency syntax
- Analysis: lots of downloadable and trainable state of the art parsers
- Tools: Google n-gram corpus viewer, grew, ...
  Syntax-based relations: https://books.google.com/ngrams
  E.g: drink=>*_NOUN
- Applications:
  - Question answering:
    What percentage of the nation's cheese does Wisconsin produce?
    The corpus contains this sentence:
    In Wisconsin, where farmers produce 28% of the nation's cheese, ...
    In the dependency tree, produce and Wisconsin are linked by an edge
  - Relation extraction:
    (MELVILLE, MOBY-DICK)
    (TOLSTOY, WAR AND PEACE)
    (MARQUÉZ, 100 YEARS OF SOLITUDE)
    (SHAKESPEARE, A MIDSUMMER NIGHT'S DREAM)
  - Sentiment analysis: There is no reason at all to believe the polluters will suddenly become reasonable

Grew: https://grew.fr/grs/parsing//

- Rule-based Dependency analyzer for French
- Analysis: allows the specification of patterns for search taking any UD corpus as input
- Graph rewriting: allows the modification, deletion and creation of new arcs:
- Online (search) and downloadable (command-line)

Grew pattern-based search: `http://match.grew.fr/`

- Rules are used to define dependency structures
- The system allows to specify lemmas, wordforms, parts of speech, dependency relations, ...
- Example: Who owns what?

  pattern {
  $VERB - [nsubj] -> SUBJ$; % VERB dominates a subject
  $VERB - [obj] -> OBJ$; % VERB dominates an object
  VERB [lemma = "own"];
  }

Assignment 2: Grew pattern-based search exercise: http://match.grew.fr/

Define rules to detect the following elements:

- Who write what:
  (*UD_English – EWT*@2.9) Darin Fisher wrote this response on January 25, 2005.
  (*UD_English – GUM*@2.9) Montalvo is holding a copy of the book Blown for Good critical of Scientology, written by Marc Headley.
- Who find what:
  (*UD_English – Atis*@2.9) could you find me the cheapest fare from boston to san francisco
- Think of a pattern(s) of your interest, write rules to detect it and present the examples found.
- Hint: to write the search patterns, it will be necessary to first draw the dependency trees using udpipe:
  http://lindat.mff.cuni.cz/services/udpipe/
- To collect results, you can use the "Export" option (select the "pivot" or element around which the given patterns will be arranged)