LING/C SC 581:
Advanced Computational Linguistics

Lecture 22

Prof. Sandiway Fong

Today's Topic

- Homework 10 Review
  - in particular, the Extra Credit part
- Follow-up to Homework 10
  - Language Acquisition
  - a look at the *Childes Database* wrt. frames of *break*
  - compare with PTB

Catalog *break* verb frames: **T1**

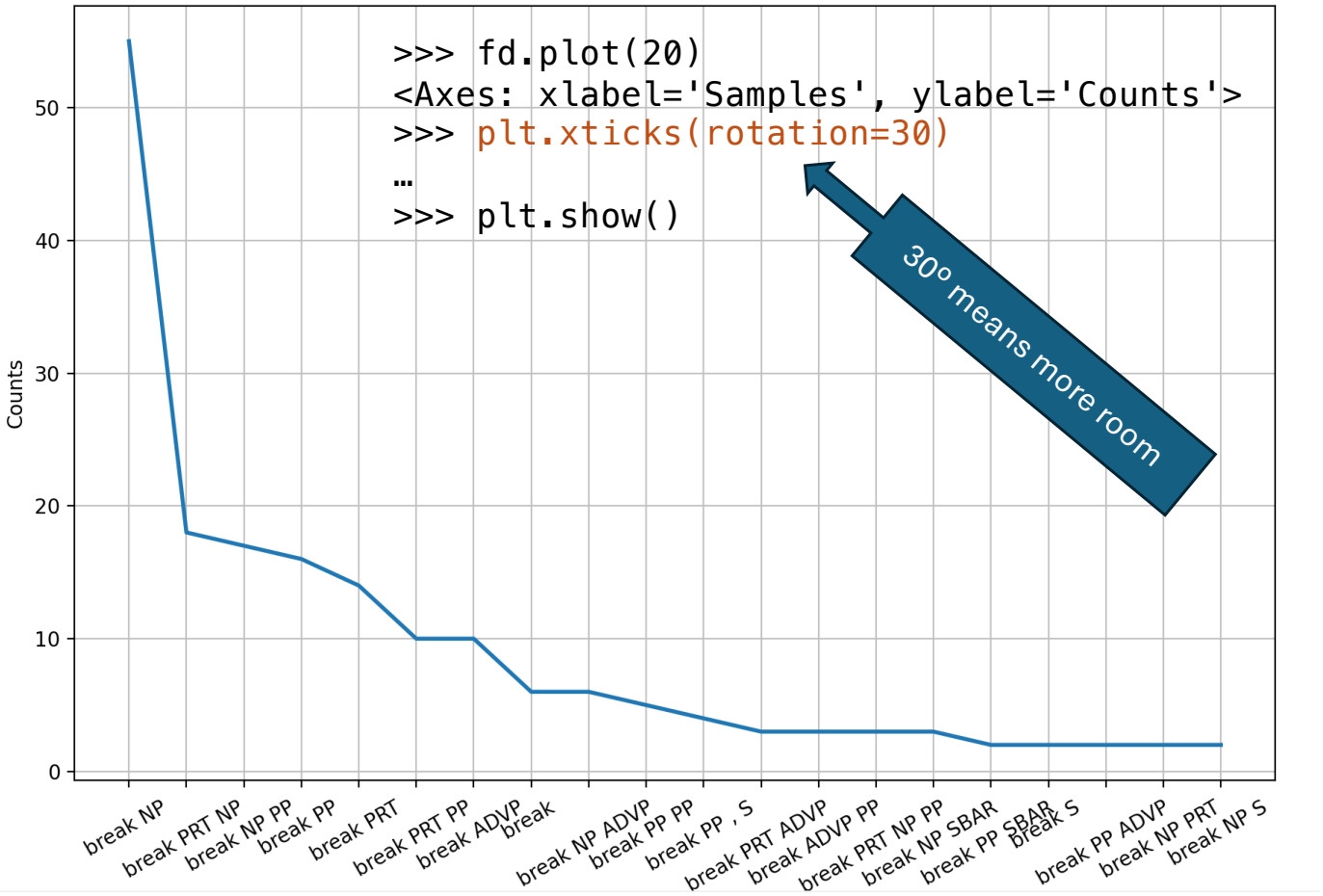
- Collect signature strings into a list, count the number of occurrences.

```
>>> break_sigs
['break ADVP SBAR', 'break NP', 'break PP SBAR', 'break PRT PP', 'break PRT', 'break NP', 'break PRT NP', 'break NP', 'break',
 'break , SBAR', 'break PRT', 'break PP', 'break PRT NP', 'break NP PP', 'break NP', 'break PRT NP', 'break PRT ADVP', 'break
ADVP PP', 'break S', 'break PP', 'break PP ADVP', 'break NP PP', 'break NP', 'break PP', 'break NP', 'break PRT PP , PP', 'VBP
CC break NP PP', 'break NP', "`` break '' CONJP VBZ SBAR", 'break PP ADVP', 'break PRT NP', 'break PRT PP ADVP', 'break ADVP',
'break ADVP', 'break NP', 'break NP PRT', 'break PP', 'break PP', 'break NP', 'break PRT NP PP', 'break NP ADVP', 'break NP S',
'break PP', 'ADVP break PRT ADVP', 'break NP ADVP', 'break NP PP', 'break NP', 'break NP', 'break PP , S', 'break PRT', 'break
PRT PP SBAR', 'break NP', 'break NP', 'break NP', 'break PP PP', 'break NP', 'break PP', 'break NP', 'break NP', 'break ADVP',
'break NP PRT', 'break PP , S', 'break PP PP', 'break NP PP', 'break S PP , VP', 'break PP PP', 'break PP', 'break NP PP',
'break NP', 'break PP NP', 'break PRT , S', 'break PRT PP', 'break PRT ADVP : `` NP', 'break ADVP', 'break NP', 'break NP PP',
'break NP S , S', 'break NP PP', 'break PRT NP', 'break PRT , S', 'break PRT PP', 'break NP PP', 'ADVP break NP', 'break NP',
'break PRT S', 'break ADVP NP PP', 'break NP', 'break NP', 'break ADVP', 'break NP ADVP', 'break PRT', 'break NP', 'break NP',
'break', 'break ADVP PP SBAR ADVP', 'break', 'break NP', 'break NP', 'break NP NP', 'break PRT NP', 'break NP', 'break PRT PP',
'break NP ADVP', 'break PP PP', 'break PP PP', 'break PRT NP , SBAR', 'break PRT NP PP', 'break PP', 'break PP , S', 'break
PP', 'break PRT NP', 'break NP SBAR', 'break NP', 'break NP', "break ADVP '' PP", 'break PP , S', 'break NP', 'break NP', 'ADVP
break NP S', 'break NP', 'break PRT PP', 'ADVP break ADVP PP', 'break NP NP', 'break PRT NP', 'break PRT', 'break PRT ADVP',
'break NP', 'break ADVP', 'break NP', 'break NP', 'break ADVP PP', 'break PRT NP PP', 'break NP SBAR', 'break ADVP', 'break NP
SBAR', 'break NP', 'break NP', 'break ADVP PP', 'break NP', 'break PRT NP', 'break PRT NP', 'break', 'ADVP break ADJP NP ,
SBAR', 'break', 'break ADVP PP PP', 'break PRT', 'break PRT NP ADVP', 'break', 'break PRT NP', 'break PRT', 'break PRT NP',
'break PRT NP', 'ADVP break ADVP PP', 'break ADVP ADVP', 'break NP', 'break NP PP', 'RB VB CC break NP', 'break PRT', 'break NP
PP', 'break PRT NP SBAR', 'break PRT', 'break NP PP', 'break NP', 'break PRT NP', 'break PP', 'break NP', 'break NP PP', 'break
PRT NP', 'break PP', 'break NP', 'break PRT NP PP , S', 'break PRT', 'break PRT NP SBAR', 'break NP , S', 'break NP', 'break
PRT', 'break PRT NP', 'break PP S', 'break PRT', 'break NP', 'break S', 'break PRT NP', 'break PP', 'break NP S', 'break PRT
PP', 'break ADVP', 'break NP ADVP', 'break NP', 'break NP PP', 'break NP', 'break NP ADVP', 'break NP', 'break NP PP', 'break
NP', 'break NP', 'break NP', 'break NP', 'break PP', 'break PRT UCP', 'break PRT', 'break NP PP', 'break ADJP PP', 'break NP',
'break PRT PP', 'break PP', 'break ADVP', 'break NP', 'ADVP break ADVP', 'break PRT ADVP', 'break NP PP', 'break PRT NP S',
'break PRT PP', 'break PRT PP', 'break PRT NP', 'break NP', 'break PP', 'break PP ADVP ADVP', 'break ADVP', 'break NP PP',
'break NP', 'break PRT PP', 'break PRT', 'break PP SBAR', 'break PRT NP ADVP PP , SBAR']
```

Catalog *break* verb frames: **T2-T3**

```
>>> import nltk
>>> fd = nltk.FreqDist(break_sigs)
>>> fd
FreqDist({'break NP': 55, 'break PRT NP': 18, 'break
 NP PP': 17, 'break PP': 16, 'break PRT': 14,
 'break PRT PP': 10, 'break ADVP': 10, 'break': 6,
 'break NP ADVP': 6, 'break PP PP': 5, ...})
>>> fd.most_common(10)
[('break NP', 55), ('break PRT NP', 18), ('break NP
 PP', 17), ('break PP', 16), ('break PRT', 14),
 ('break PRT PP', 10), ('break ADVP', 10),
 ('break', 6), ('break NP ADVP', 6), ('break PP
 PP', 5)]
```

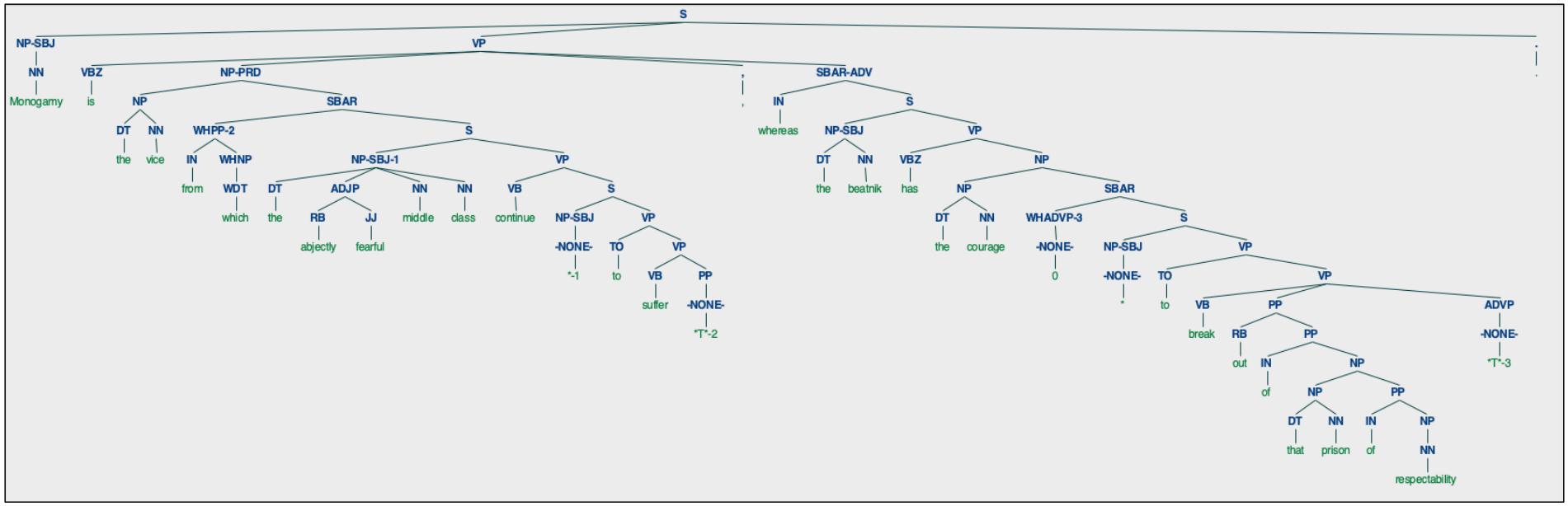
Catalog break verb frames:
T4



Find all *break* verb frames

```
23  break/VB PP *T*-3/ADVP
>>> break_trees[23].draw()
```

what is this?



Homework 10: Part 2

- Extra Credit Task 5:
  - Use the Stanza parser in nltk to parse the PTB *break* sentences
  - Compare the distribution of verb frame signature strings to that obtained from the PTB.

Homework 10: Part 2

- **Workflow**:

```
$ python3 -i hw10ec.py
13 (s), sents: 244
93 (s), stanza_trees: 244
s_fs: 236
s_vfs: 216
s_sigs: 216
```

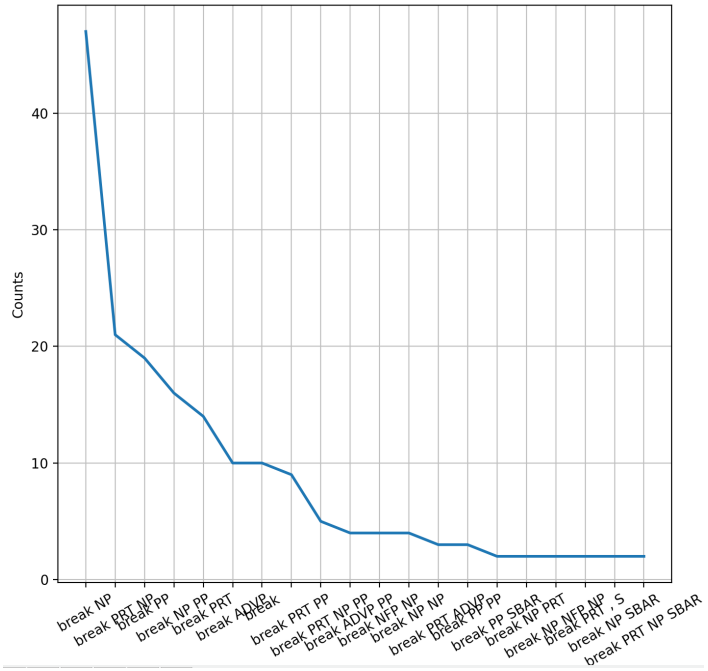
initialize pipeline, extracts sentences
parse sentences
get all frames
limit to verb frames only
convert to signatures

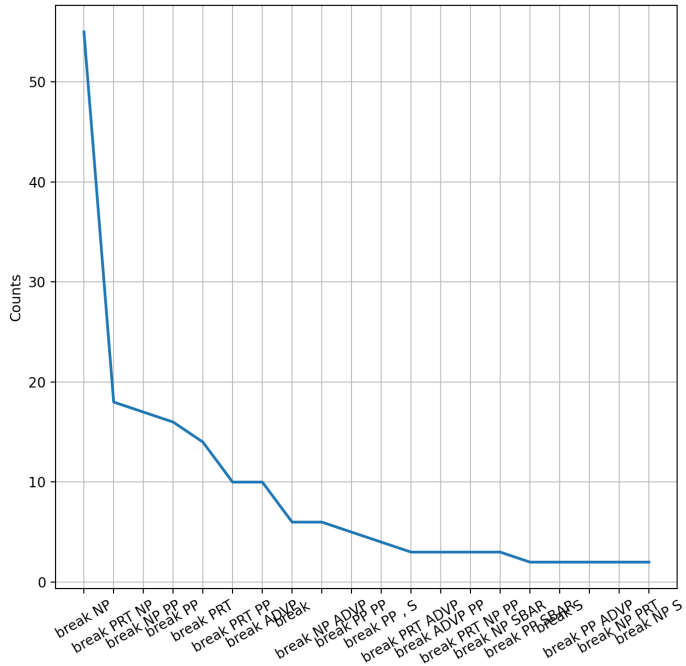
```
>>> s_sigs
['break PP SBAR', 'break NP', 'break PP PP S', 'break ADVP PP', 'break PRT', 'break NP', 'break PRT NP', 'break NP',
 'break NFP NP', 'break PRT', 'break PP', 'break PRT NP', 'break NP PP', 'break NP', 'break PRT NP', 'break PRT NP',
 'break ADVP', 'break NP S', 'break PP', 'break PP NFP NP', 'break NP', 'break PP', 'break NP', 'break PRT PP , PP',
 'VBP CC break NP PP', 'break NP', 'break PP ADVP', 'break PRT NP', 'break PRT PP', 'break NFP NP', 'break ADVP', 'b
reak NP', 'break NP PRT', 'break PP', 'break PP', 'break NP', 'break PRT NP PP', 'break NP NFP NP', 'break', 'break
```

Homework 10: Part 2

```
>>> s_sigs
['break PP SBAR', 'break NP', 'break PP PP S', 'break ADVP PP', 'break PRT', 'break NP', 'break PRT NP', 'break NP',
 'break NFP NP', 'break PRT', 'break PP', 'break PRT NP', 'break NP PP', 'break NP', 'break PRT NP', 'break PRT NP',
 'break ADVP', 'break NP S', 'break PP', 'break PP NFP NP', 'break NP', 'break PP', 'break NP', 'break PRT PP , PP', 'VBP
 CC break NP PP', 'break NP', 'break PP ADVP', 'break PRT NP', 'break PRT PP', 'break NFP NP', 'break ADVP', 'break NP',
 'break NP PRT', 'break PP', 'break PP', 'break NP', 'break PRT NP PP', 'break NP NFP NP', 'break', 'break PP', 'break
 PRT ADVP', 'break NP NFP NNP CD', 'break NP PP', 'break NP PP', 'break', 'break PP , SYM S', 'break ADVP', 'break PRT PP
 SBAR', 'break NP', 'break NP', 'break NFP NP', 'break PP PP', 'break NP', 'break', 'break PRT PP', 'break NP', 'break
 NP', 'break PRT', 'break NP PRT', 'break PP', 'break PP , PP NFP PP , PP', 'break NP PP', 'break NP PP , S', 'break PP
 PP', 'break PP', 'break NP PP', 'break NP', 'break PP NP', 'break PRT , S', 'break PRT PP', 'break PRT NP', 'break PP',
 'break NP', 'break NFP PP', 'break NP NP , S', 'break NP PP', 'break PRT NP', 'break PRT , S', 'break PRT PP', 'break NP
 PP', 'break NP', 'break NP', 'break PRT NP', 'break PRT NP NFP NP', 'break NP PP', 'break', 'break NP', 'break NP NP',
 'break PRT', 'break NP', 'break NP', "`` break '' ''", 'break ADVP PP S', 'break PP', 'break NP', 'break NP NP', 'break
 PRT NP', 'break NP', 'break PRT PP', 'break NP ADJP', 'break PP PP', 'break PRT NP , SBAR', 'break PRT NP PP', 'break
 PP', 'break PP', 'break PP', 'break PRT NP', 'break NP', 'break NP', 'break ADVP', 'break PP , NFP S', 'break NP',
 'break NP', 'break NP NP S', 'break NP', 'break ADVP PP', 'break ADVP PP', 'break NP NP', 'break PRT NP', 'break PRT',
 'break PRT ADVP', 'break NP', 'break ADVP', 'break NP', 'break NP', 'break PRT NP PP', 'break NP SBAR', 'break ADVP',
 'break NP SBAR', 'break NP', 'break NP NFP NP', 'break ADVP PP', 'break NP', 'break PRT NP', 'break PRT NP', 'break',
 'break ADVP NP , SBAR', 'break', 'break ADVP PP PP', 'break PRT', 'break PRT NP ADVP', 'break', 'break PRT NP', 'break
 PRT', 'break PRT NP', 'break PRT NP', 'break ADVP', 'break NP', 'break NP PP', 'VB CC break NP', 'break PRT', 'break NFP
 PP NFP NP', 'break PRT NP SBAR', 'break PRT', 'break NP PP', 'break NP', 'break PRT NP', 'break PP', 'break NP NP',
 'break NP PP', 'break PRT NP', 'break PP', 'break NFP NP', 'break PRT NP PP', 'break PRT', 'break PRT NP SBAR', 'break
 NP , NFP S', 'break', 'break PRT', 'break PRT NP', 'break PP S', 'break PRT', 'break NP', 'break ADVP', 'break PRT NP',
 'break PP', 'break PRT PP', 'break ADVP', 'break NP PP', 'break NP', 'break NP PP', 'break NP', 'break NP ADVP', 'break
 NP', 'break', 'break NP', 'break', 'break NP', 'break NP', 'break PP', 'break PRT SBAR : NP', 'break PRT', 'break NP
 PP', 'break ADJP PP', 'break NP', 'break PRT PP', 'break PP', '`` break ADVP CC VP', 'break NP', 'break ADVP', 'break
 PRT ADVP', 'break NP PP', 'break PRT NP NP S', 'break PRT NFP NP', 'break PRT PP', 'break PRT NP', 'break NP', 'break
 PP', 'break PP ADVP NFP NP', 'break NP', 'break NP PP', 'break NP', 'break PRT PP', 'break PRT', 'break PP SBAR', 'break
 PRT NP PP']
```

Homework 10: Part 2





Homework 10: Part 2

```
>>> fd = nltk.FreqDist(s_sigs)
>>> fd
FreqDist({'break NP': 47, 'break PRT
  NP': 21, 'break PP': 19, 'break NP
  PP': 16, 'break PRT': 14, 'break
  ADVP': 10, 'break': 10, 'break PRT
  PP': 9, 'break PRT NP PP': 5,
  'break ADVP PP': 4, ...})
>>> fd.most_common(10)
[('break NP', 47), ('break PRT NP',
  21), ('break PP', 19), ('break NP
  PP', 16), ('break PRT', 14),
  ('break ADVP', 10), ('break', 10),
  ('break PRT PP', 9), ('break PRT
  NP PP', 5), ('break ADVP PP', 4)]
```

```
>>> fd = nltk.FreqDist(break_sigs)
>>> fd
FreqDist({'break NP': 55, 'break PRT
  NP': 18, 'break NP PP': 17, 'break
  PP': 16, 'break PRT': 14, 'break
  PRT PP': 10, 'break ADVP': 10,
  'break': 6, 'break NP ADVP': 6,
  'break PP PP': 5, ...})
>>> fd.most_common(10)
[('break NP', 55), ('break PRT NP',
  18), ('break NP PP', 17), ('break
  PP', 16), ('break PRT', 14),
  ('break PRT PP', 10), ('break
  ADVP', 10), ('break', 6), ('break
  NP ADVP', 6), ('break PP PP', 5)]
```

Language Acquisition

- What exposure do kids get for verb frames for *break* ?
  - Compared to the variety attested in the **PTB**?
    - Compared to that in **WordNet**
      - Compared to that in the **OED**
- **General Q**:
  - how do we acquire the variety of verb frames we see?
  - let's take a look at the *Childes Database* wrt. frames of *break*
  - and compare with PTB

Revisiting the verb *break*

- Big unanswered question:
  - why does *break* have so many different senses?

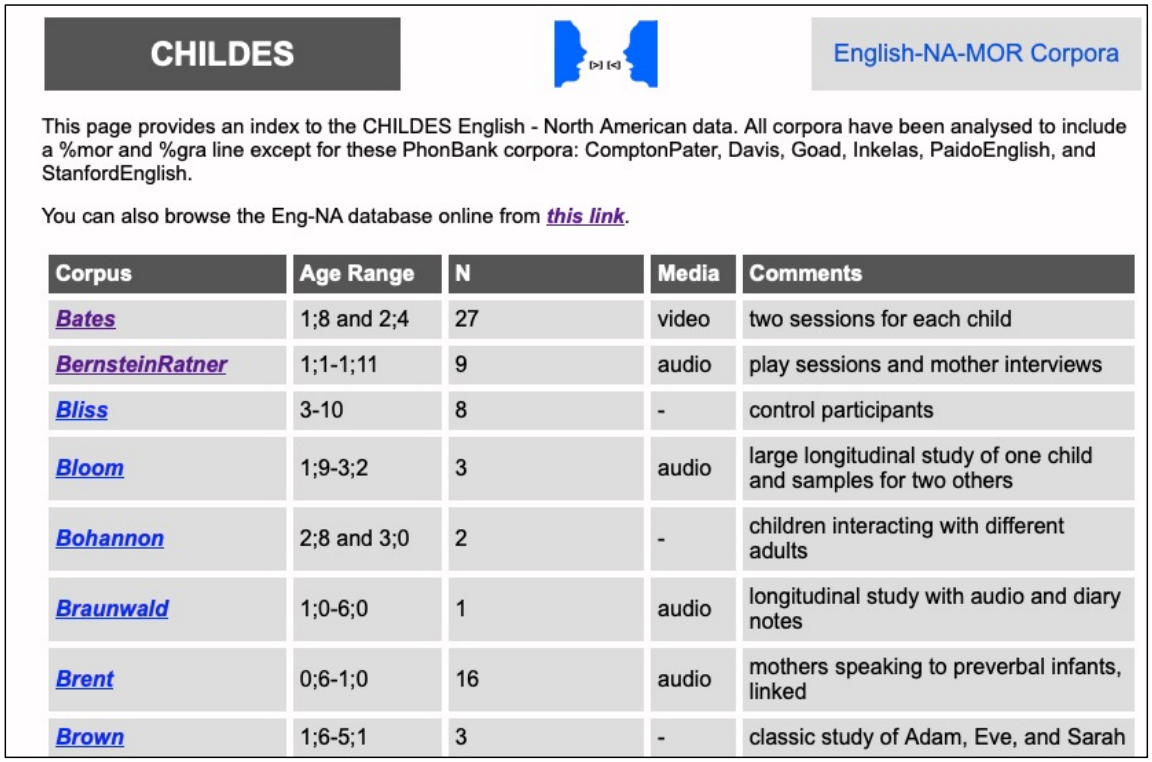
Is the data out there rich?

- ptb corpus:
  - ptb = Brown + Wall Street Journal (1.7m words)
  - 73K sentences: 244 with *break* used in a verb frame (VP)
  - 121 different verb frames
- Childes database:
  - non-parsed: we'll parse them
  - 231 with *break* used in a verb frame
  - 22 different verb frames

We can compare the `ptb` to what children hear or produce

Childes Database

- https://childes.talkbank.org/access/Eng-NA/



CHILDES English-NA-MOR Corpora

This page provides an index to the CHILDES English - North American data. All corpora have been analysed to include a %mor and %gra line except for these PhonBank corpora: ComptonPater, Davis, Goad, Inkelas, PaidoEnglish, and StanfordEnglish.

You can also browse the Eng-NA database online from *this link*.

Corpus	Age Range	N	Media	Comments
Bates	1;8 and 2;4	27	video	two sessions for each child
BernsteinRatner	1;1-1;11	9	audio	play sessions and mother interviews
Bliss	3-10	8	-	control participants
Bloom	1;9-3;2	3	audio	large longitudinal study of one child and samples for two others
Bohannon	2;8 and 3;0	2	-	children interacting with different adults
Braunwald	1;0-6;0	1	audio	longitudinal study with audio and diary notes
Brent	0;6-1;0	16	audio	mothers speaking to preverbal infants, linked
Brown	1;6-5;1	3	-	classic study of Adam, Eve, and Sarah

Childes Database

Data is unparsed, but we want to find verb frames!
Note: similarly, HW10 EC Part

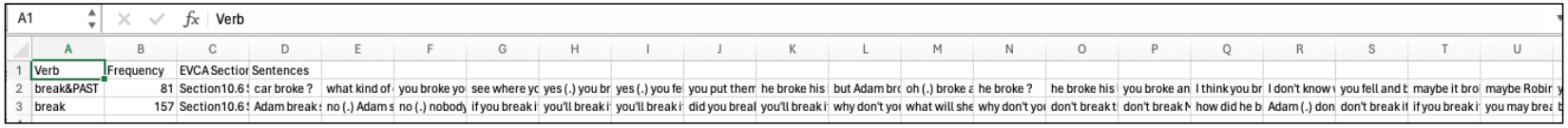
```
@Languages: eng
@Participants: CHI Amy Target_Child , MOT Mother
@ID: eng|Bates|CHI|2;04.|female|TD|MC|Target_Child|||
@ID: eng|Bates|MOT||female|||Mother|||
@Types: cross, toyplay, TD
1    MOT: c(o)m(e) on over here and sit down .
2    CHI: uhuh .
3    CHI: no .
4    MOT: uhhuh .
5    MOT: put it up here (.) on the table .
6    CHI: &-uh [?] in here !
7    MOT: no (.) we shouldn't eat that in the living room .
8    CHI: uhuh eat it [*] .
9    MOT: you want to eat it right there ?
10   CHI: (o)kay ?
```

- *Break*:

break&PAST,81,Section10.6 Section40.8.3 Section45.1 Section48.1.1 , car broke ?, what kind of car broke ?, you broke your horn ., see where you broke your horn ?, yes (.) you broke it ., yes (.) you fell down and broke your horn (.) didn't you ?, you put them in your bank (.) remember and it broke your bank ?, he broke his bank and he's showing it to you ., but Adam broke the wagon (.) didn't he ?, oh (.) broke a pencil ., he broke ?, he broke his leg ?, you broke another wheel ., I think you broke it ., I don't know why you broke it ., you fell and broke your head ., maybe it broke (.) Adam ., maybe Robin broke that one ., you broke another one ?, you broke it (.) didn't you ?, you broke that thing off ?, you broke it ., yes (.) you broke it ., because you broke it .

Childes Database

File: `break_childes.csv`



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	Verb	Frequency	EVCA Sectior	Sentences																	
2	break&PAST	81	Section10.6	car broke ?	what kind of	you broke yo	see where yc	yes (.) you br	yes (.) you fe	you put them	he broke his	but Adam bro	oh (.) broke a	he broke ?	he broke his	you broke an	I think you br	I don't know	you fell and b	maybe it bro	maybe Robir
3	break	157	Section10.6	Adam break	no (.) Adam s	no (.) nobody	if you break i	you'll break i	you'll break i	did you breal	you'll break i	why don't yo	what will she	why don't yo	don't break t	don't break N	how did he b	Adam (.) don	don't break it	if you break i	you may brea

- `preprocess()`: grab examples from csv file

```
86# childes corpus tools¶
87¶
88def preprocess(file='break_childes.csv'):¶
89    raw = open(file, encoding='utf-8').read()¶
90    lines = raw.splitlines()¶
91    # line 0 is header line: ignore¶
92    # Verb,Frequency,EVCA Sections,Sentences¶
93    sents = []¶
94    for line in lines[1:]:¶
95        # break&PAST,81,Section10.6 Section40.8.3 Section45.1 Section48.1.1 …
  ¶
96        sents.extend([s for s in line.split(',') if s != ''][3:])¶
97    return sents¶
```

Childes Database

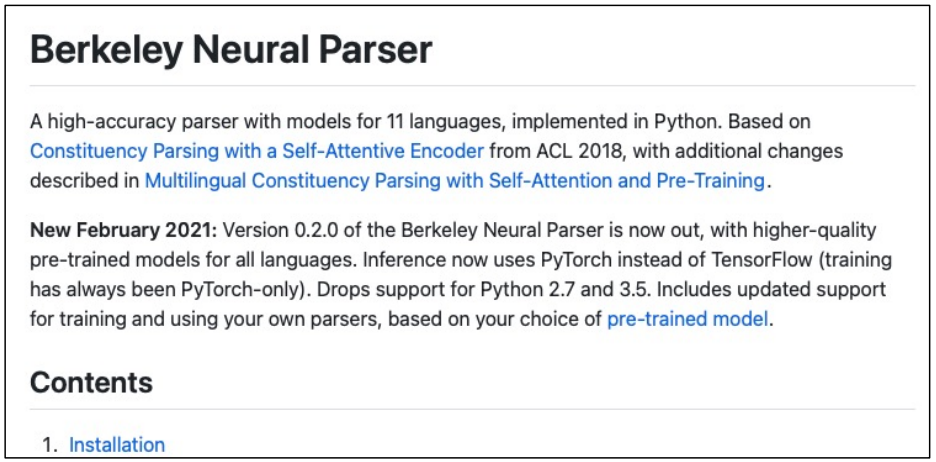
```
>>> sents
```

```
[' car broke ?', ' what kind of car broke ?', ' you broke your horn .', ' see where you broke your horn ?', ' yes (.) you broke it .', " yes (.) you fell down and broke your horn (.) didn't you ?", ' you put them in your bank (.) remember and it broke your bank ?', " he broke his bank and he's showing it to you .", " but Adam broke the wagon (.) didn't he ?", ' oh (.) broke a pencil .', ' he broke ?', ' he broke his leg ?', ' you broke another wheel .', ' I think you broke it .', " I don't know why you broke it .", ' you fell and broke your head .', ' maybe it broke (.) Adam .', ' maybe Robin broke that one .', ' you broke another one ?', " you broke it (.) didn't you ?", ' you broke that thing off ?', ' you broke it .', ' yes (.) you broke it .', ' because you broke it .', " I don't know why he broke his leg .", ' you broke the boat .', ' I think you broke it .', ' the milk broke too .', " I think you broke these off the horse (.) didn't you ?", ' Adam (.) I think Mommy broke the dolly .', ' broke the leg off .', ' Jack fell down and broke his crown .', " I don't know who broke it .", ' it broke ?', ' it broke ?', ' it looks like it broke .', ' she looked at this yesterday and kept saying (.) â\x80\x9cshoe brokeâ\x80\x9d +...', " yes (.) â\x80¡ captain's boot broke .", ' yes (.) â\x80¡ the ping+pong broke .', ' it broke ?', " Pap'll have to fix the box (.) (be)cause it broke .", ' you broke it .', ' remember when you broke my glasses ?', " I broke my cream pitcher so I'll just give it to you .", ' it broke (.) yeah .', ' &hmm ?', ' &hmm ?', ' you broke it (.) yeah .', ' right there (.) see where you broke it .', ' who broke that ?', ' see (.) he sat on the house and he broke it all down .', ' you broke it .', " I don't know why you broke it (.) you broke it .", … ' if you bounce it (.) you break it .', " you'll break it .", " no (.) you'll break your neck carryin(g) that .", " Sarah (.) don't break [?] .", " you're gonna break your neck ."]
```

Constituency Parser

- We could use Stanza with `nltk` (*as in HW10 EC*)
  or we could use in `nltk` / spaCy the *Berkeley Neural Parser*

  https://github.com/nikitakit/self-attentive-parser



Berkeley Neural Parser

A high-accuracy parser with models for 11 languages, implemented in Python. Based on Constituency Parsing with a Self-Attentive Encoder from ACL 2018, with additional changes described in Multilingual Constituency Parsing with Self-Attention and Pre-Training.

New February 2021: Version 0.2.0 of the Berkeley Neural Parser is now out, with higher-quality pre-trained models for all languages. Inference now uses PyTorch instead of TensorFlow (training has always been PyTorch-only). Drops support for Python 2.7 and 3.5. Includes updated support for training and using your own parsers, based on your choice of pre-trained model.

Contents

1. Installation

benepar

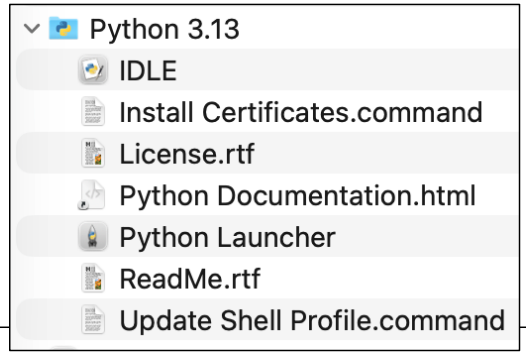
```
$ pip3 install benepar
Collecting benepar
  Downloading benepar-0.2.0.tar.gz (33 kB)
Requirement already satisfied: benepar in
/Library/Frameworks/Python.framework/Versions/3.13/lib/python3.13/site-packages (0.2.0)
Collecting spacy>=2.0.9 (from benepar)
Collecting torch>=1.6.0 (from benepar)
Collecting torch-struct>=0.5 (from benepar)
Collecting tokenizers>=0.9.4 (from benepar)
Collecting transformers>=4.2.2 (from transformers[tokenizers,torch]>=4.2.2->benepar)
Collecting protobuf (from benepar)
Collecting sentencepiece>=0.1.91 (from benepar)
…
Successfully installed MarkupSafe-2.1.5 accelerate-0.29.3 annotated-types-0.6.0 benepar-0.2.0 blis-0.7.11 catalogue-
2.0.10 certifi-2024.2.2 charset-normalizer-3.3.2 cloudpathlib-0.16.0 confection-0.1.4 cymem-2.0.8 filelock-3.13.4
fsspec-2024.3.1 huggingface-hub-0.22.2 idna-3.7 jinja2-3.1.3 langcodes-3.4.0 language-data-1.2.0 marisa-trie-1.1.0
mpmath-1.3.0 murmurhash-1.0.10 networkx-3.3 packaging-24.0 preshed-3.0.9 protobuf-5.26.1 psutil-5.9.8 pydantic-2.7.1
pydantic-core-2.18.2 pyyaml-6.0.1 requests-2.31.0 safetensors-0.4.3 sentencepiece-0.2.0 setuptools-69.5.1 smart-
open-6.4.0 spacy-3.7.4 spacy-legacy-3.0.12 spacy-loggers-1.0.5 srsly-2.4.8 sympy-1.12 thinc-8.2.3 tokenizers-0.19.1
torch-2.3.0 torch-struct-0.5 transformers-4.40.1 typer-0.9.4 typing-extensions-4.11.0 urllib3-2.2.1 wasabi-1.1.2
weasel-0.3.4
```

benepar

```
>>> import benepar
>>> benepar.download('benepar_en3')
[nltk_data] Error loading benepar_en3: <urlopen error [SSL:
[nltk_data]     CERTIFICATE_VERIFY_FAILED] certificate verify failed:
[nltk_data]     unable to get local issuer certificate (_ssl.c:1018)>
False
```

works on one mac I have but not the other!

macOS: /Applications/Python 3.13




```
~$ /Applications/Python\ 3.13/Install\ Certificates.command ; exit;
 -- pip install --upgrade certifi
Requirement already satisfied: certifi in /Library/Frameworks/Python.framework/Versions/3.13/lib/python3.13/site-
packages (2024.12.14)
Collecting certifi
  Downloading certifi-2026.2.25-py3-none-any.whl.metadata (2.5 kB)
Downloading certifi-2026.2.25-py3-none-any.whl (153 kB)
Installing collected packages: certifi
  Attempting uninstall: certifi
    Found existing installation: certifi 2024.12.14
    Uninstalling certifi-2024.12.14:
      Successfully uninstalled certifi-2024.12.14
Successfully installed certifi-2026.2.25
```

benepar vs stanza

```
import nltk
import benepar
# benepar.download('benepar_en3')
import spacy
nlp = spacy.load('en_core_web_md')
nlp.add_pipe("benepar", config={"model":
"benepar_en3"})
doc = nlp(sents)
sent = list(doc.sents)[0]
tree =
nltk.Tree.fromstring(sent._.parse_string)
```

```
import nltk
import stanza
# stanza.download('en')
nlp = stanza.Pipeline(lang='en',
processors='tokenize,pos,constituen
cy')
nlp(sent).
sentences[0].constituency
```

spaCy: benepar usage

Usage with spaCy (recommended)

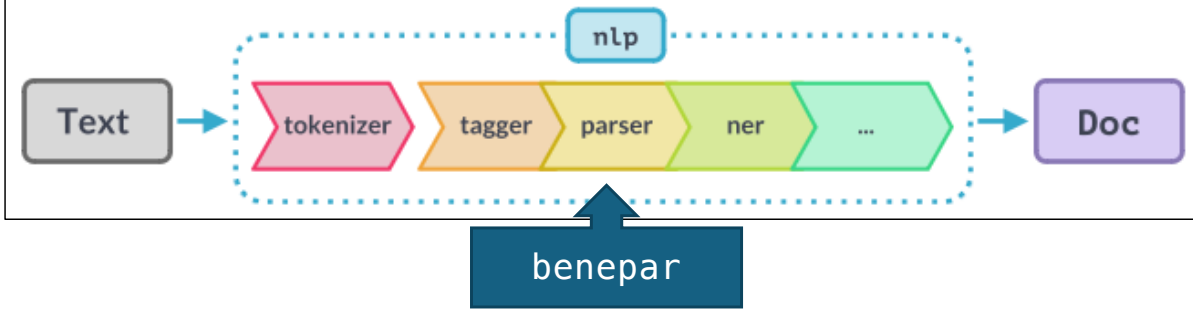
The recommended way of using benepar is through its integration with spaCy:

```
>>> import benepar, spacy
>>> nlp = spacy.load('en_core_web_md')
>>> if spacy.__version__.startswith('2'):
        nlp.add_pipe(benepar.BeneparComponent("benepar_en3"))
    else:
        nlp.add_pipe("benepar", config={"model": "benepar_en3"})
>>> doc = nlp("The time for action is now. It's never too late to do something.")
>>> sent = list(doc.sents)[0]
>>> print(sent._.parse_string)
(S (NP (NP (DT The) (NN time)) (PP (IN for) (NP (NN action)))) (VP (VBZ is) (ADVP (R
>>> sent._.labels
('S',)
>>> list(sent._.children)[0]
The time for action
```

spaCy pipeline

- Installation via pip: https://spacy.io/usage

When you call `nlp` on a text, spaCy first tokenizes the text to produce a `Doc` object. The `Doc` is then processed in several different steps – this is also referred to as the **processing pipeline**. The pipeline used by the trained pipelines typically include a tagger, a lemmatizer, a parser and an entity recognizer. Each pipeline component returns the processed `Doc`, which is then passed on to the next component.



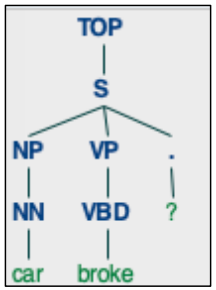
benepar

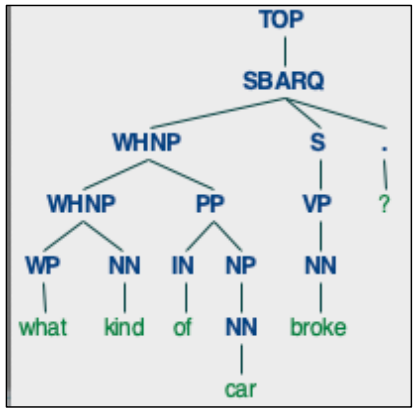
```
$ python3 -i break_childes.py
verbforms: {'broken', 'breaks', 'breaking',
   'break', 'broke'}
loaded Childes data: 238 sentences
benepar failed for << you broke your horn .>>
benepar failed for << you put them in your bank (.)
   remember and it broke your bank ?>>
benepar failed for << I think you broke it .>>
benepar failed for << you fell and broke your head
   .>>
benepar failed for << because you broke it .>>
benepar failed for << I think you broke it .>>
benepar failed for << I think you broke these off
   the horse (.) didn't you ?>>
benepar failed for << broke the leg off .>>
benepar failed for << she looked at this yesterday
   and kept saying (.) â□□shoe brokeâ□□ +...>>
benepar failed for << I broke my cream pitcher so
   I'll just give it to you .>>
benepar failed for << who broke that ?>>
benepar failed for << I broke it .>>
benepar failed for << who broke it ?>>
benepar failed for << broke her front tooth .>>
benepar failed for << you know you broke it .>>
benepar failed for << broke it .>>
benepar failed for << who broke it ?>>
benepar failed for << broke her heart .>>
benepar failed for << (a)n(d) you didn't break any
   (.) huh ?>>
Benepar VP frames (benepar_vfs): 215
fd2: Childes verb frame distribution: total: 215,
   types: 16
```

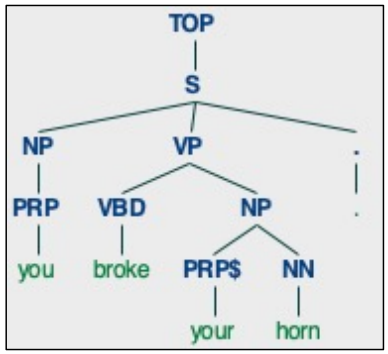
Childes Database: `nltk` tokenization 1

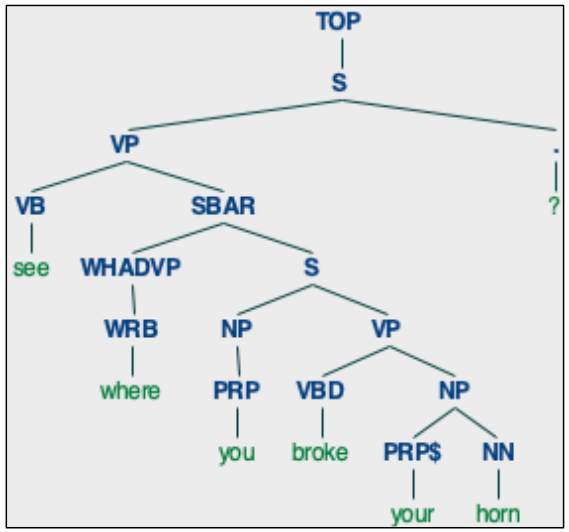
```
>>> wtl = tokenize(preprocess())
>>> len(wtl)
236
```

```
parser = benepar.Parser("benepar_en3")
>>> for wt in wtl[:10]:
...     parse(wt).draw()
```







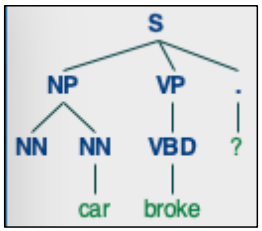


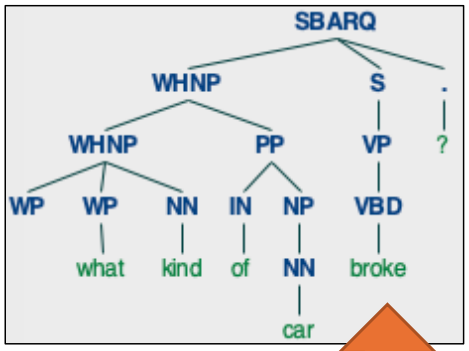
no subj, no obj under S
broke verb is NN!

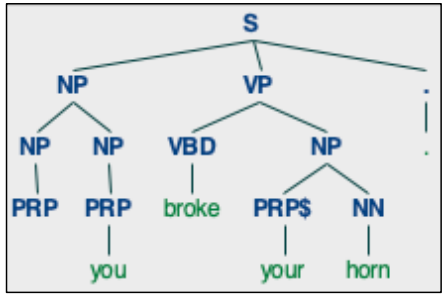
Childes Database: spacy tokenization 1

```
>>> sents = preprocess()
>>> for s in sents:
...     Tree.fromstring(sparse(s)).draw()
```

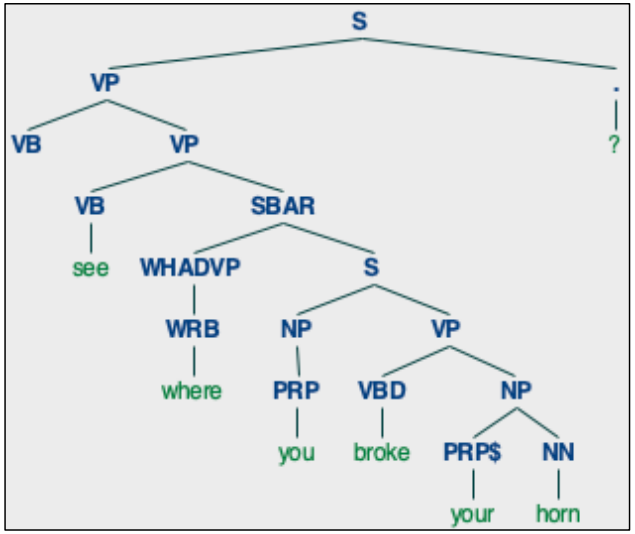
```
112# spaCy parse
113def sparse(s):
114    doc = nlp(s)
115    for d in doc.sents:
116        if find_break(d._.parse_string):
117            return d._.parse_string
```



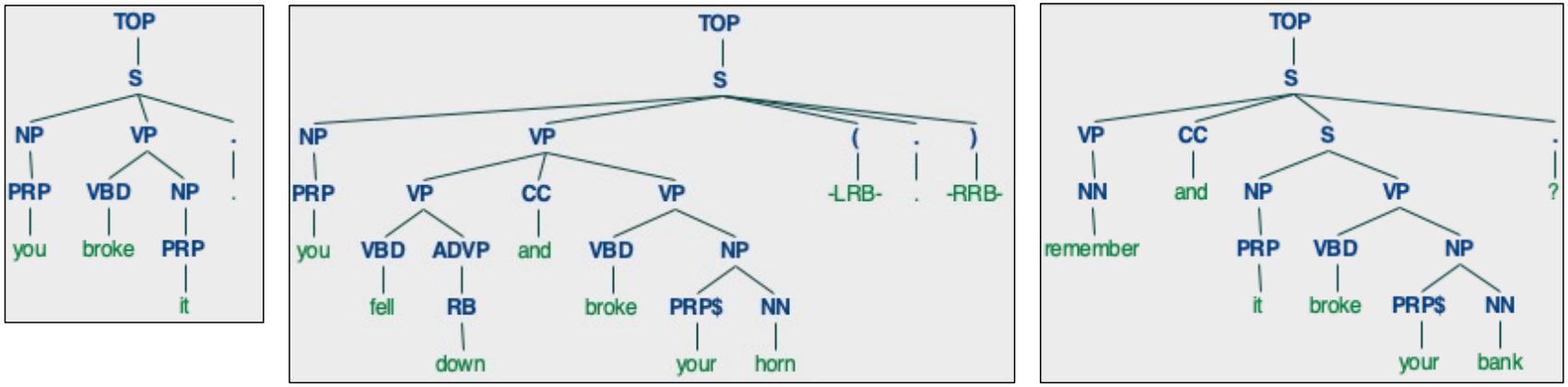




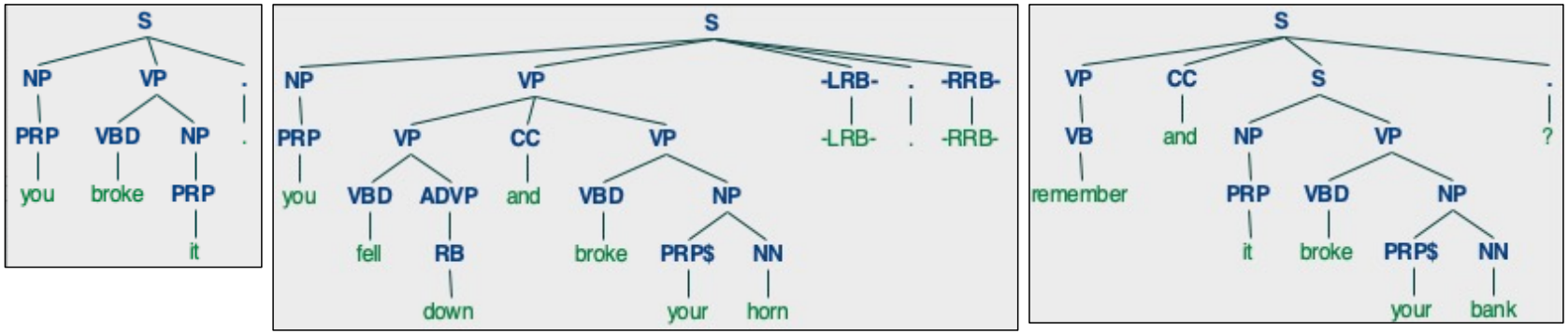
no subj, no obj under S
but verb is VBD now!
spaCy is better than nltk



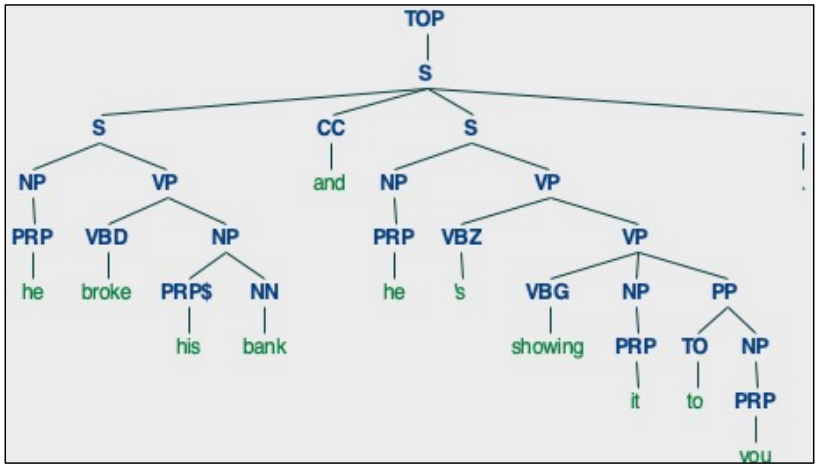
Childes Database: `nltk` tokenization 2

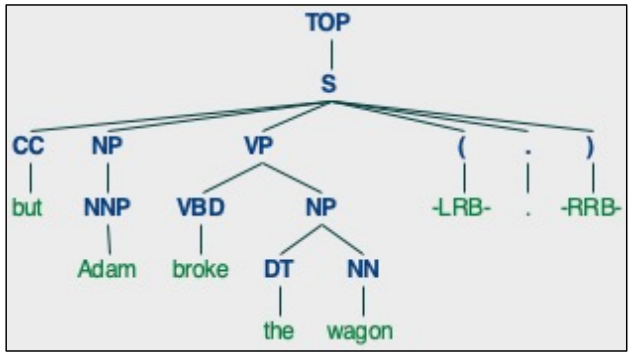


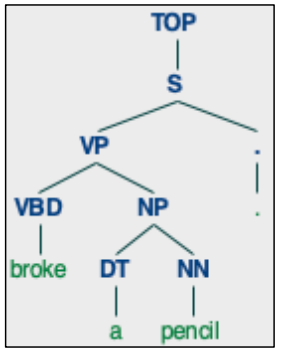
Childes Database: spacy tokenization 2



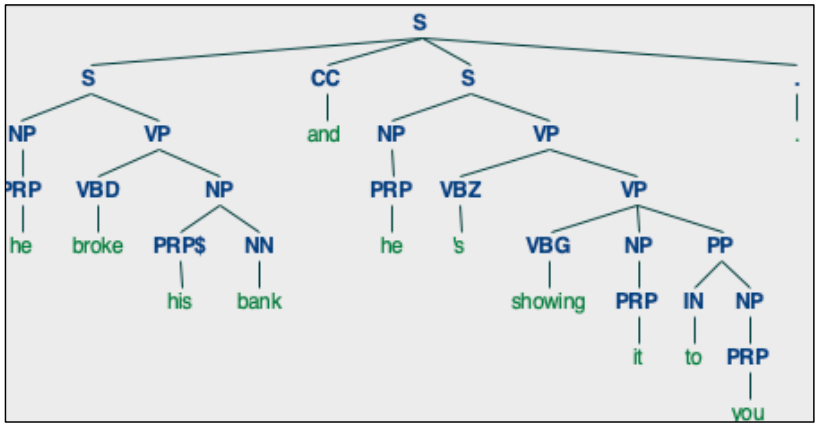
Childes Database: `nltk` tokenization 3

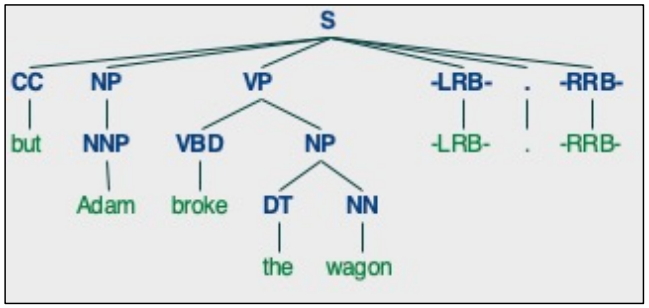


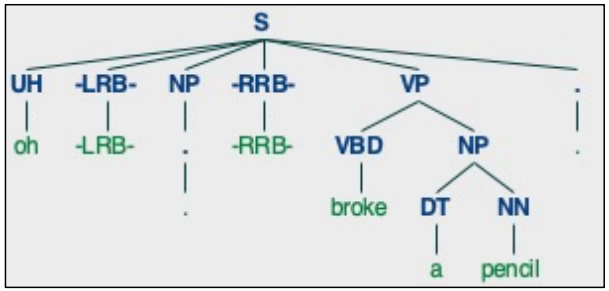


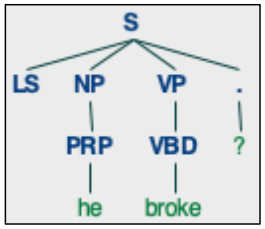


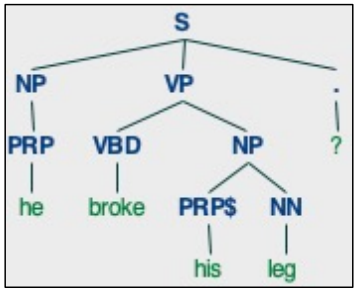
Childes Database: spacy tokenization 3

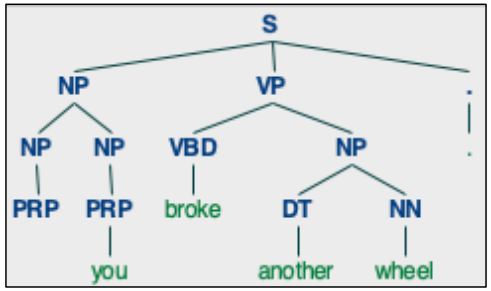




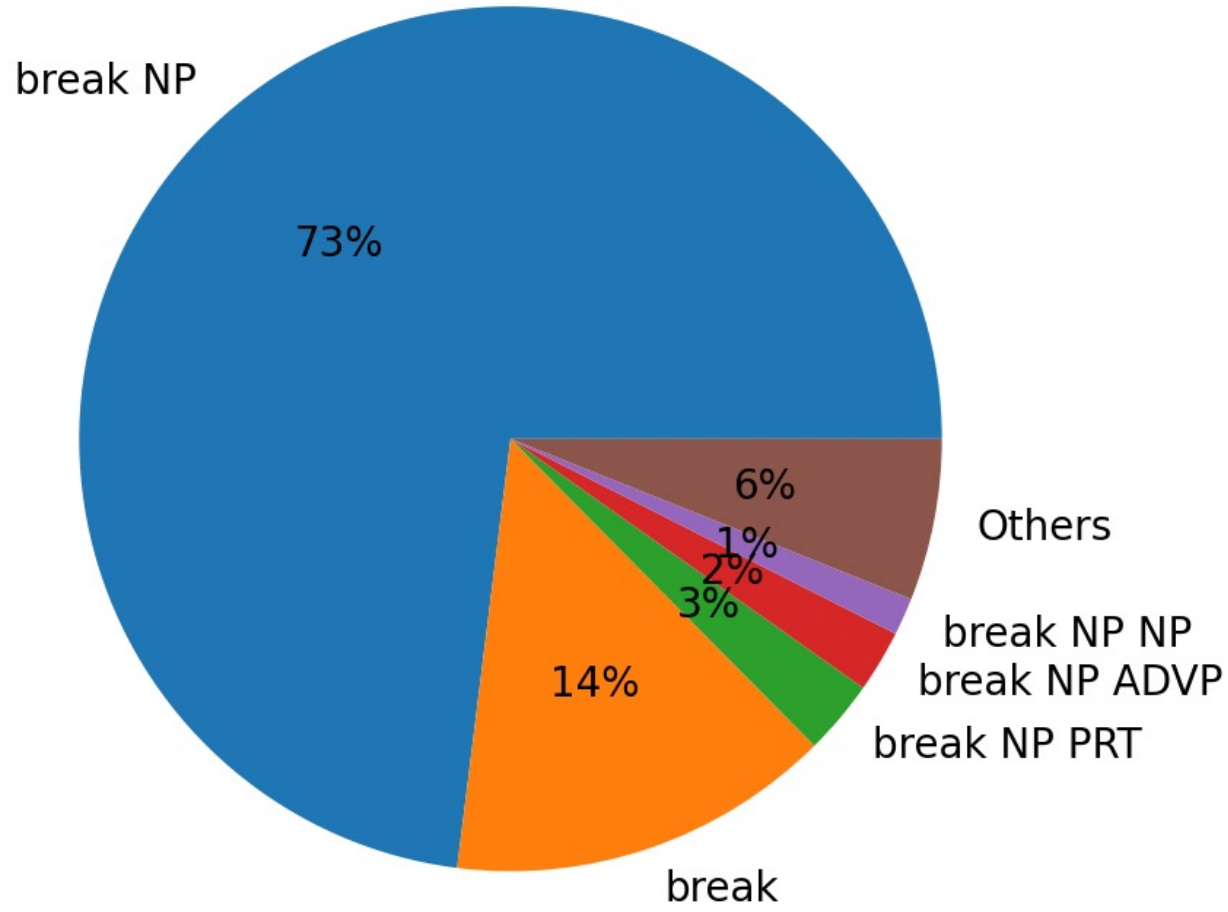






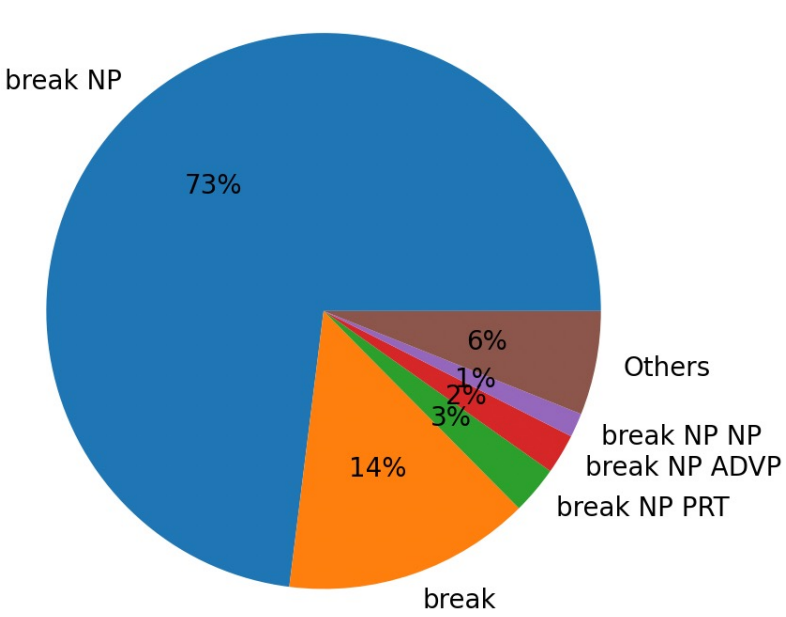


Childes Database: frame distribution

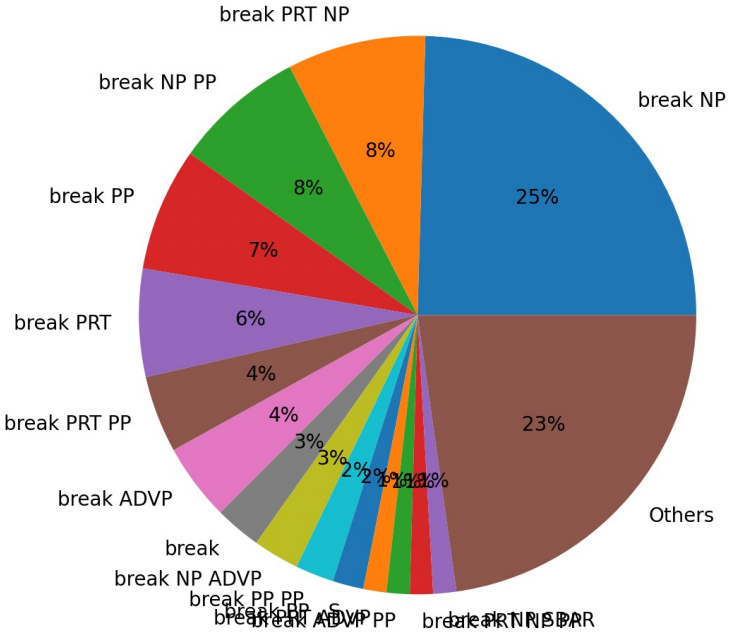


Break verb frames

Childes database



ptb



What other verbs? Class discussion