LING/C SC 581:
Advanced Computational Linguistics

Lecture 28

Today's Topic

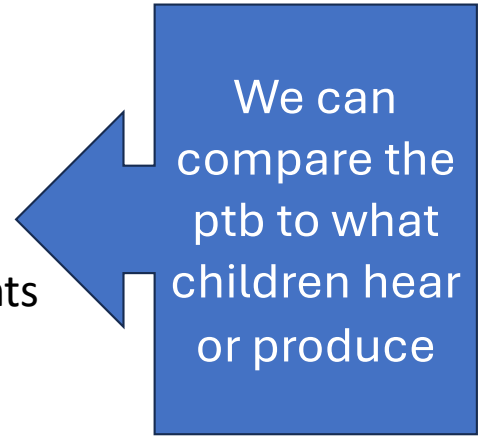
- Example of an experimental project using Python and nltk:
  - a "pipeline":
    1. get corpus data
    2. split into sentences
    3. tokenize sentences: spaCy vs. nltk
    4. parse sentences (*using the Berkeley Neural Parser*)
    5. extract verb frames (and *compare*)

Revisiting the verb *break*

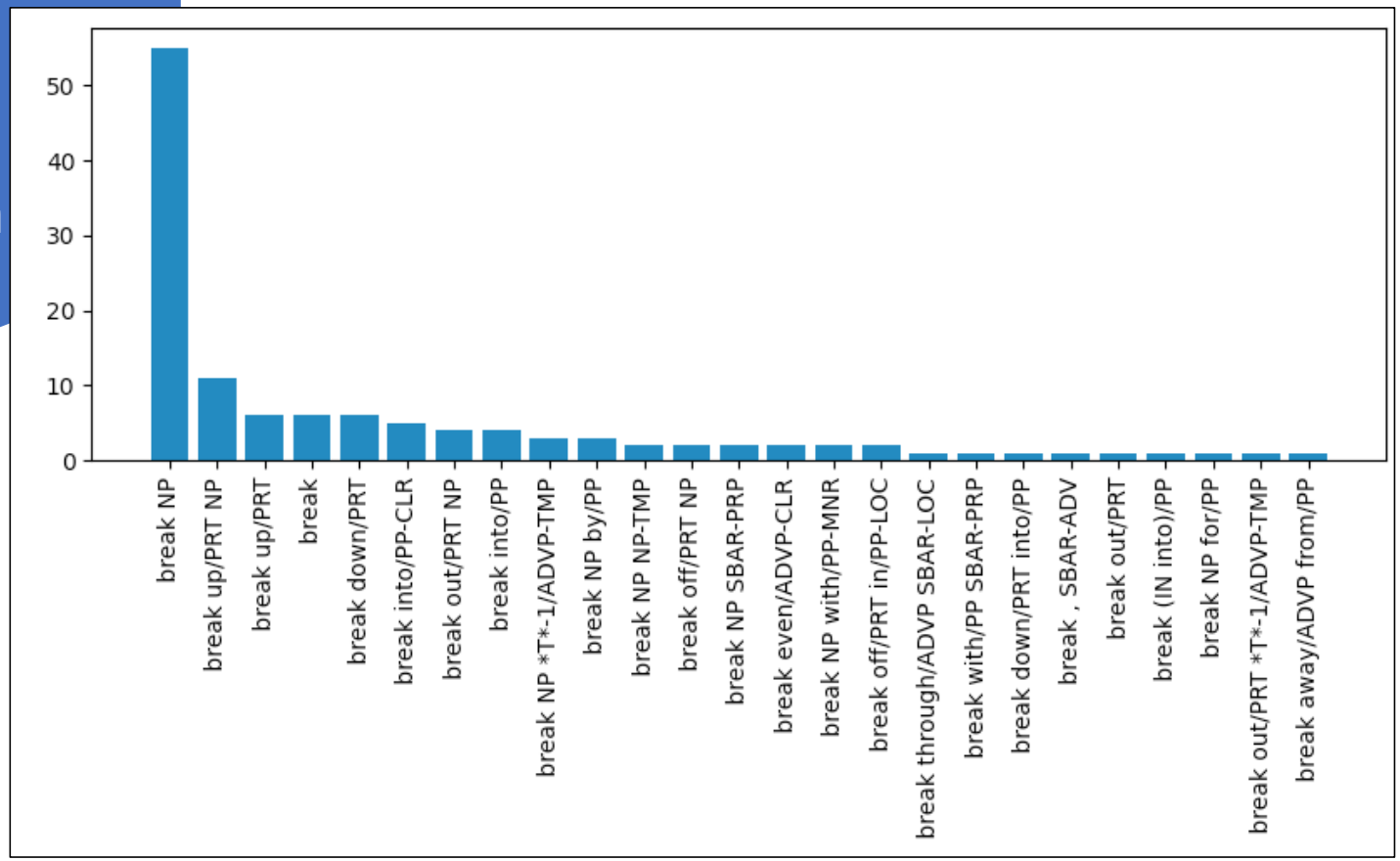
- Big unanswered question:
  - why does *break* have so many different senses?

Is the data out there rich?

- ptb corpus:
  - ptb = Brown + Wall Street Journal (1,740,895 words)
  - 73.451 sentences: 244 with *break* used in a verb frame (VP)
  - 121 different verb frames
- Childes database:
  - non-parsed: we'll parse them using a parser that builds constituents
  - 231 with *break* used in a verb frame
  - 22 different verb frames

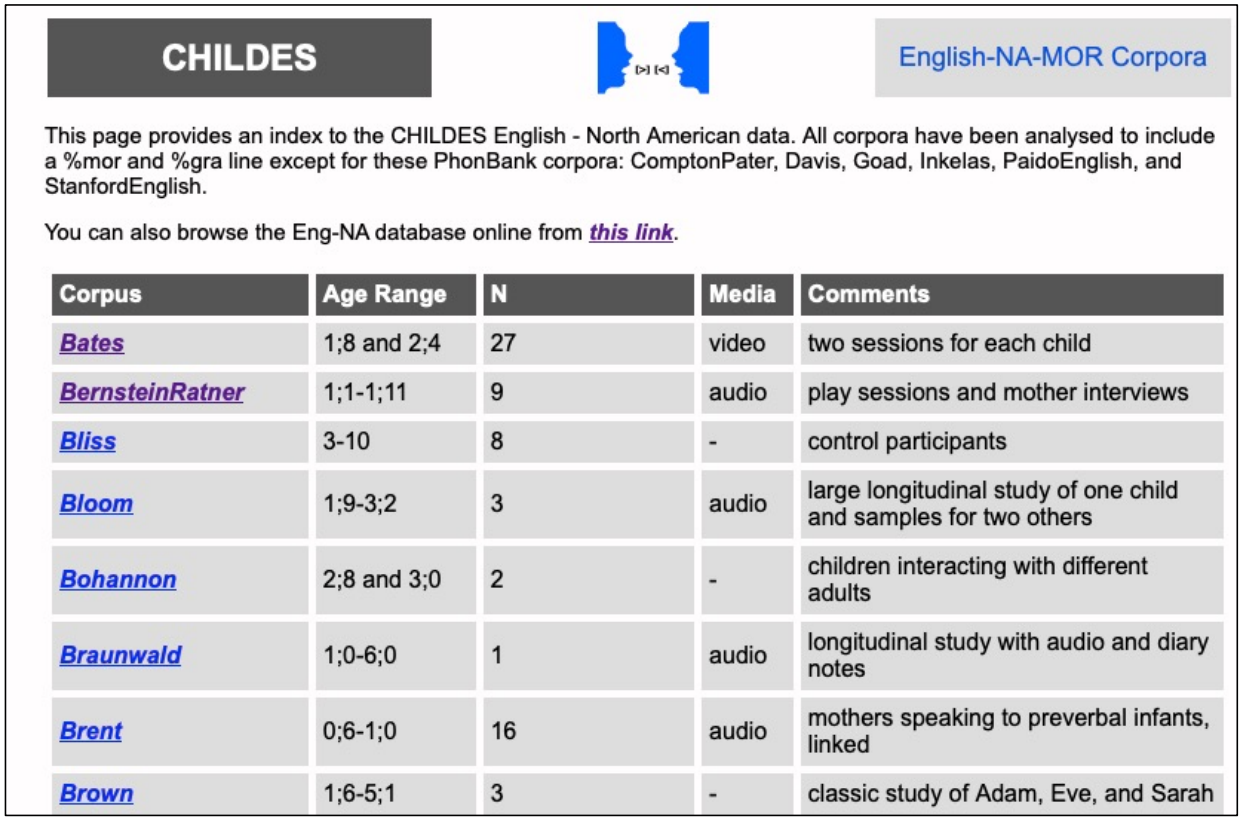


ptb database: *break* verb frame distribution



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!

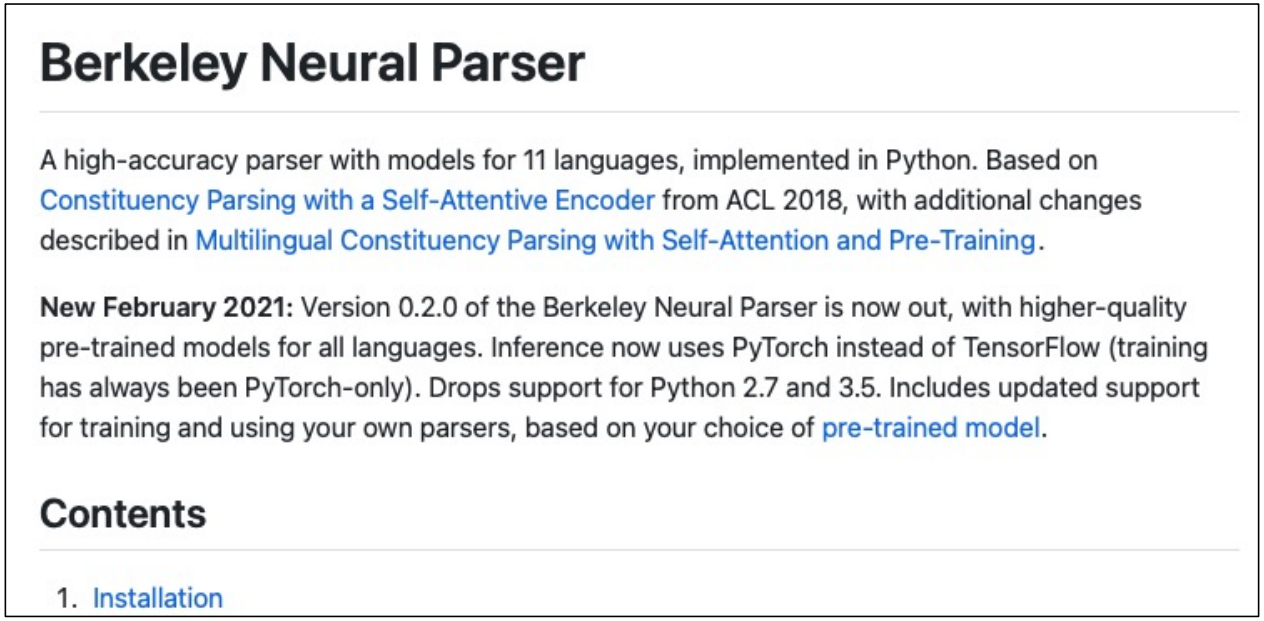
```
@Languages: eng
@Participants: CHI Amy Target_Child , MOT Mother
@ID: eng|Bates|CHI|2;04.|female|TD|MC|Target_Child|||
@ID: eng|Bates|MOT||female|||Mother|||
@Types: cross, toyplay, TD
1    MOT: c(o)m(e) on over here and sit down .
2    CHI: uhuh .
3    CHI: no .
4    MOT: uhhuh .
5    MOT: put it 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 .

benepar

- We can install this into Python (*so we call it directly from* nltk)
- 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

```
$ python -m pip install benepar
Collecting benepar
  Downloading benepar-0.2.0.tar.gz (33 kB)
Requirement already satisfied: nltk>=3.2 in ./venv/lib/python3.12/site-packages (from benepar)
(3.8.1)
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

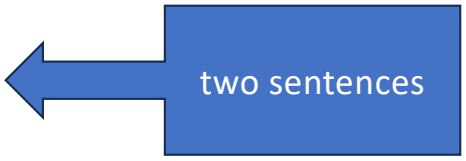
Usage with NLTK

There is also an NLTK interface, which is designed for use with pre-tokenized datasets and treebanks, or when integrating the parser into an NLP pipeline that already performs (at minimum) tokenization and sentence splitting. For parsing starting with raw text, it is **strongly encouraged** that you use spaCy and `benepar.BeneparComponent` instead.

```
>>> import benepar
>>> parser = benepar.Parser("benepar_en3")
>>> input_sentence = benepar.InputSentence(
    words=['"', 'Fly', 'safely', '.', '"'],
    space_after=[False, True, False, False, False],
    tags=['``', 'VB', 'RB', '.', "''"],
    escaped_words=['``', 'Fly', 'safely', '.', "''"],
)
>>> tree = parser.parse(input_sentence)
>>> print(tree)
(TOP (S (`` ``) (VP (VB Fly) (ADVP (RB safely))) (. .) ('' '')))
```

benepar

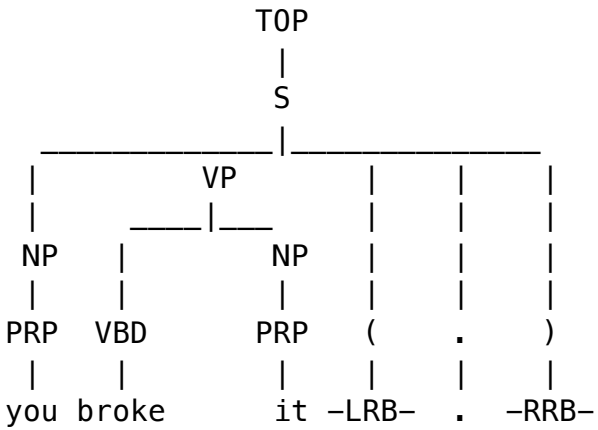
```
$ python3
Python 3.12.3 (main, Apr  9 2024, 08:09:14)
>>> import benepar
>>> benepar.download('benepar_en3')
[nltk_data] Downloading package benepar_en3 to
[nltk_data]     /Users/sandiway/nltk_data...
[nltk_data]   Unzipping models/benepar_en3.zip
…
>>> parser = benepar.Parser("benepar_en3")
>>> import nltk
>>> nltk.sent_tokenize(" you broke it (.) yeah .")
[' you broke it (.)', 'yeah .']
```



nltk: benepar usage

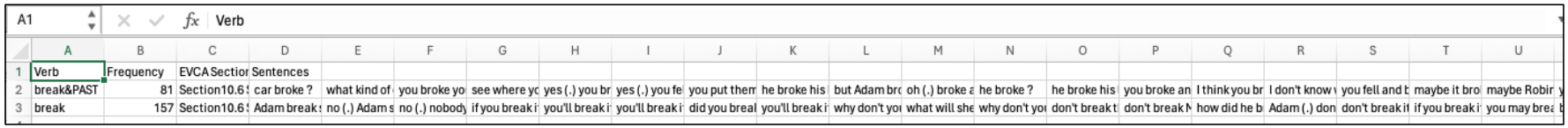
- Grab the 1st sentence out of two:

```
>>> wt = nltk.pos_tag(nltk.word_tokenize(nltk.sent_tokenize(" you broke it (.) yeah .")[0]))
>>> wt
[('you', 'PRP'), ('broke', 'VBD'), ('it', 'PRP'), ('(', '('), ('.', '.'), (')', ')')]
>>> input = benepar.InputSentence(words=[t[0] for t in wt],tags=[t[1] for t in wt])
>>> parser.parse(input).pretty_print()
```



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

- Code:
  - `verbforms = set(['break', 'breaks', 'broke', 'breaking', 'broken'])`
  - `verbtags = set(['VB', 'VBP', 'VBZ', 'VBD', 'VBG', 'VBN'])`

- `tokenize()`

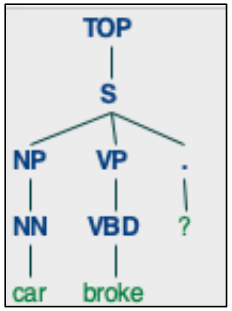
```
108# not all sentences will contain our verb, filter them out
109def tokenize(sents):
110    wtl = []
111    for s in sents:
112        rl = [sent for sent in nltk.sent_tokenize(s) if find_break(sent)]
113        for relevant in rl:
114            wtl.append(nltk.pos_tag(nltk.word_tokenize(relevant)))
115    return wtl
```

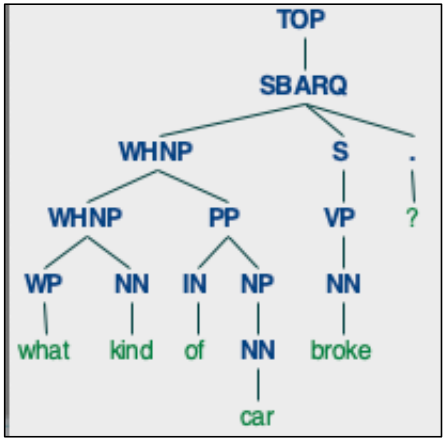
```
 99# find some verbform
100def find_break(sent):
101    found = False
102    for v in verbforms:
103        if v in sent:
104            found = True
105            break
106    return found
```

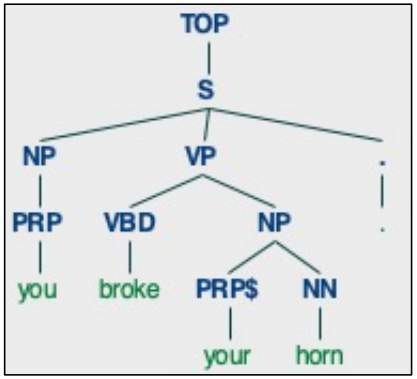
Childes Database

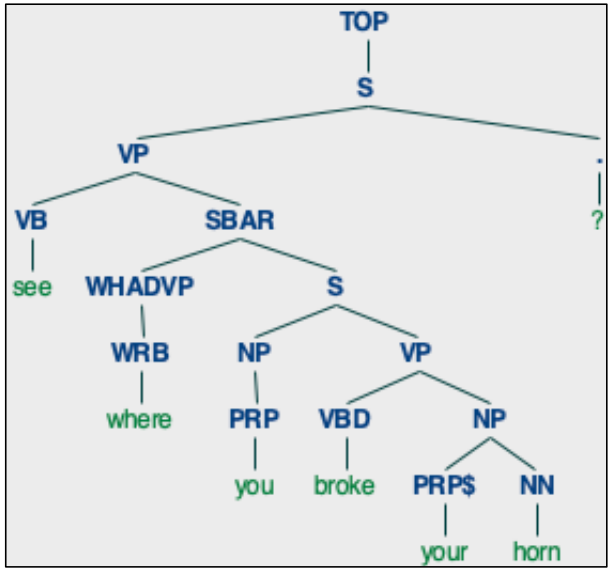
```
>>> wtl = tokenize(preprocess())
>>> len(wtl)
236
```

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



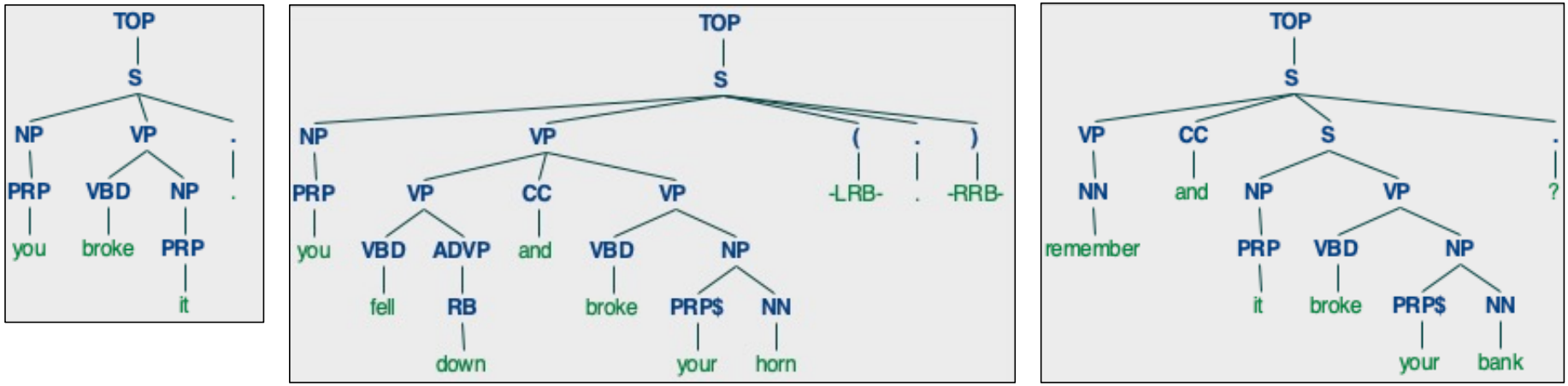




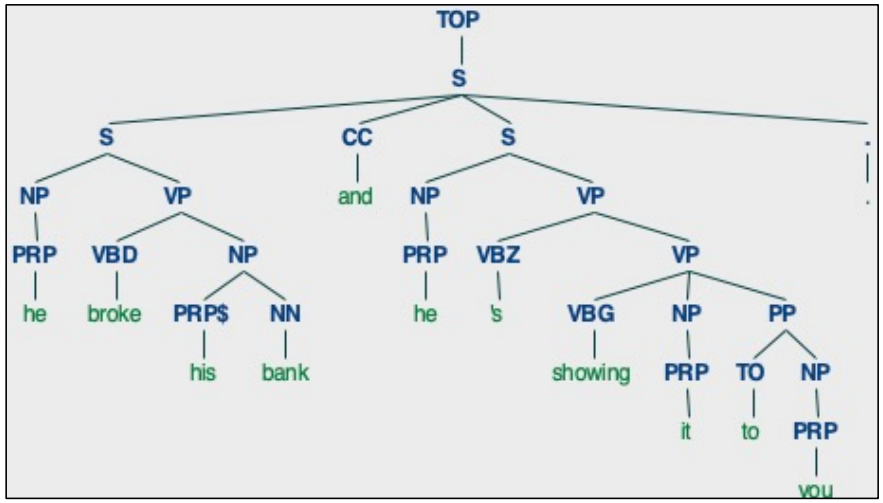


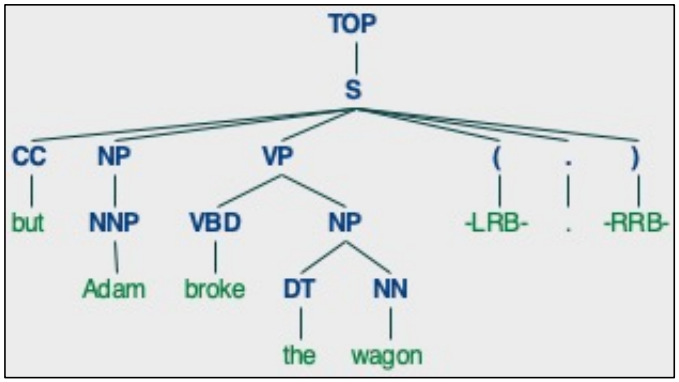
no subj, no obj under S
broke verb is NN!

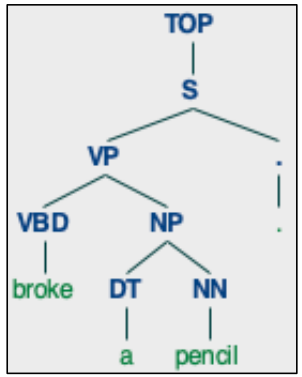
Childes Database



Childes Database







spaCy: benepar usage

Usage with spaCy (recommended)

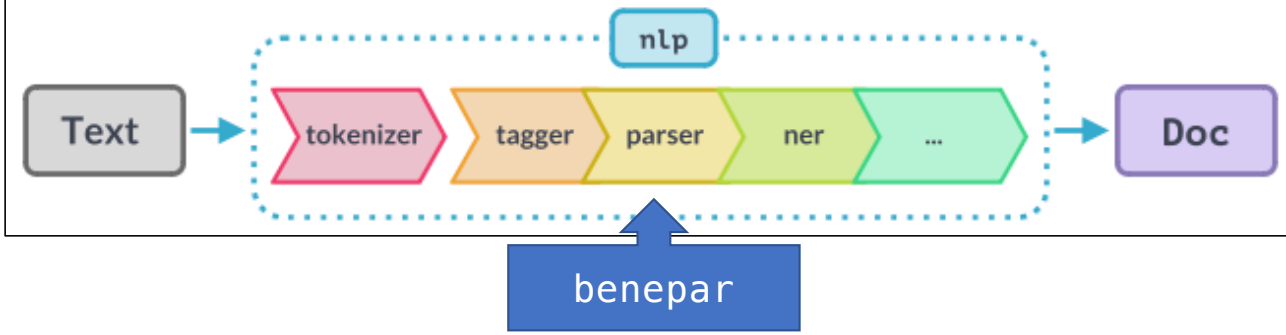
The recommended way of using benepar is through its integration with spaCy:

```
>>> 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.



spaCy English language model

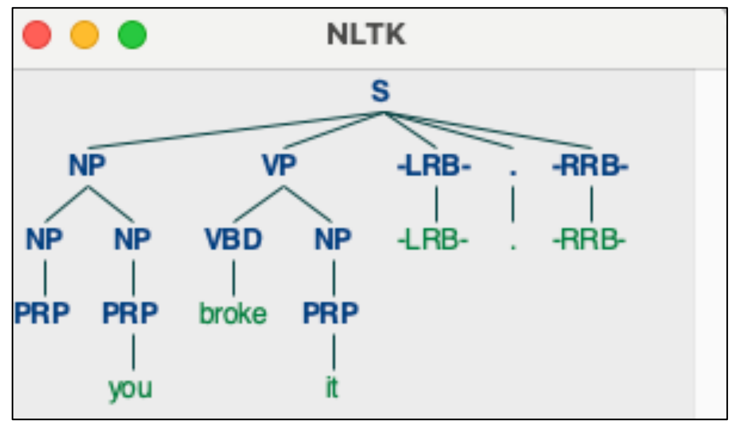
```
>>> import spacy
>>> nlp = spacy.load('en_core_web_md')
Traceback (most recent call last):
…
OSError: [E050] Can't find model 'en_core_web_md'. It doesn't seem to be a Python package
or a valid path to a data directory.
$ python3 -m spacy download en_core_web_md
Collecting en-core-web-md==3.7.1
  Downloading https://github.com/explosion/spacy-models/releases/download/en_core_web_md-
3.7.1/en_core_web_md-3.7.1-py3-none-any.whl (42.8 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 42.8/42.8 MB 7.0 MB/s eta 0:00:00
…
Installing collected packages: en-core-web-md
Successfully installed en-core-web-md-3.7.1
✔ Download and installation successful
You can now load the package via spacy.load('en_core_web_md')
```

spaCy: benepar usage

```
$ python3
>>> import benepar
>>> import spacy
>>> nlp = spacy.load('en_core_web_md')
>>> nlp.add_pipe("benepar", config={"model": "benepar_en3"})
<benepar.integrations.spacy_plugin.BeneparComponent object at 0x17394bf50>
>>> doc = nlp(" you broke it (.) yeah .")
>>> sent = list(doc.sents)[0]
>>> sent._
<spacy.tokens.underscore.Underscore object at 0x100b22e70>
>>> sent._.parse_string
'(S (NP (NP (PRP  )) (NP (PRP you))) (VP (VBD broke) (NP (PRP it))) (-LRB-
-LRB-) (. .) (-RRB- -RRB-))'
```

spaCy: benepar usage

```
>>> import nltk
>>> nltk.Tree.fromstring(sent._.parse_string).pretty_print()
                             S
         ____________________|________________
        NP                VP          |   |    |
    ____|____          ____|____      |   |    |
   NP        NP       |         NP    |   |    |
   |         |        |         |     |   |    |
  PRP       PRP      VBD       PRP  -LRB- .  -RRB-
   |         |        |         |     |   |    |
  ...       you     broke       it  -LRB- .  -RRB-
```

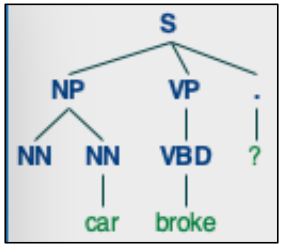


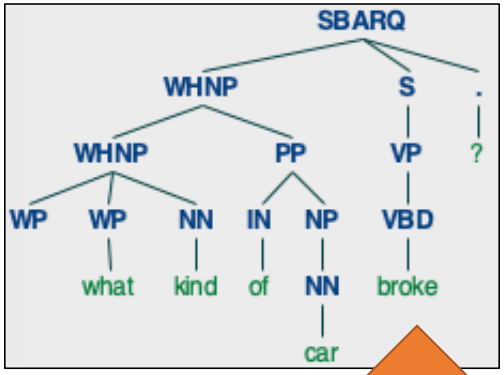

```
>>> nltk.Tree.fromstring(sent._.parse_string).draw()
>>> list(doc.sents)[1]._.parse_string
'(INTJ (UH yeah) (. .))'
```

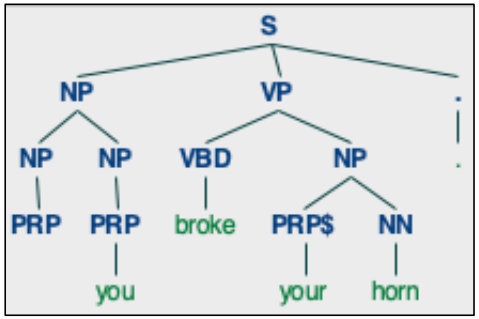
Childes Database

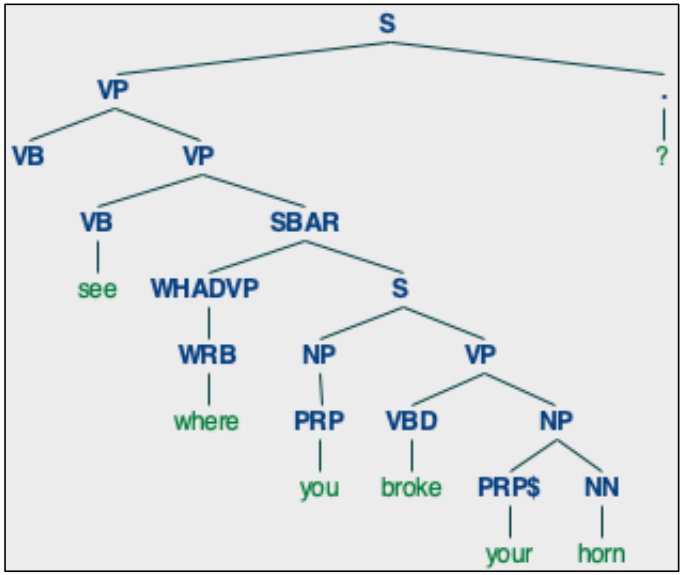
```
>>> sents = preprocess()
>>> for s in sents:
...     Tree.fromstring(sparse(s)).draw()
```

```
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¶
```



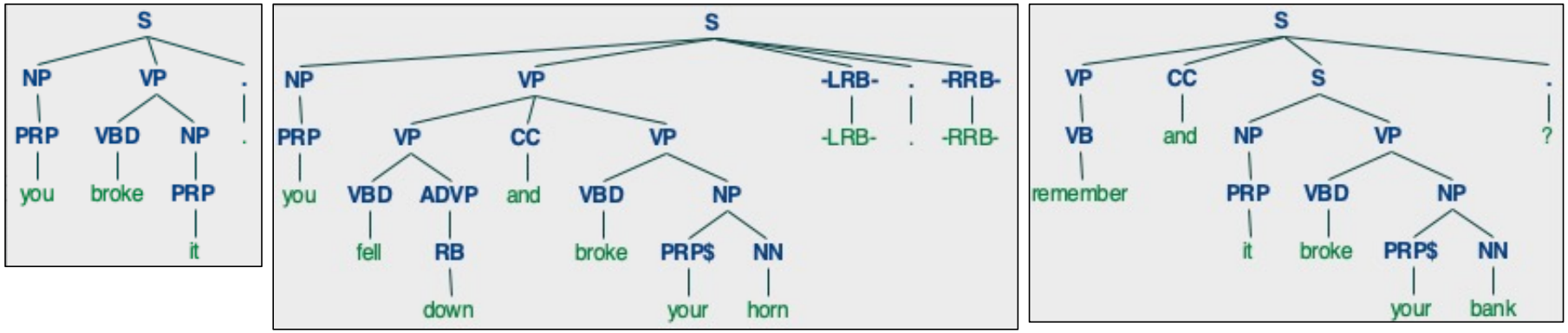




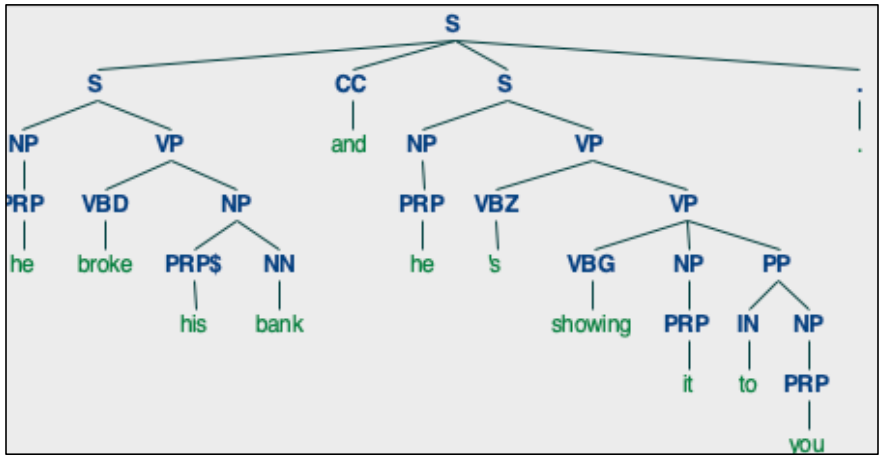


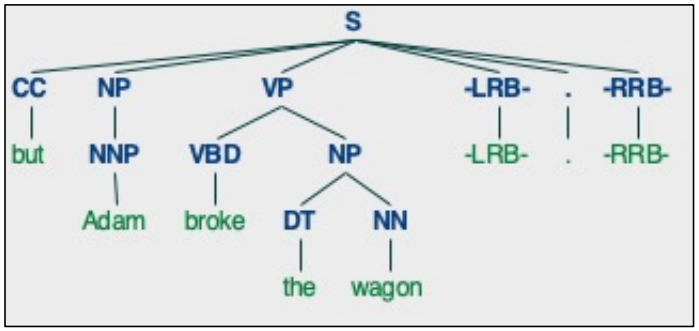
no subj, no obj under S
but verb is VBD now!
spaCy is better than nltk

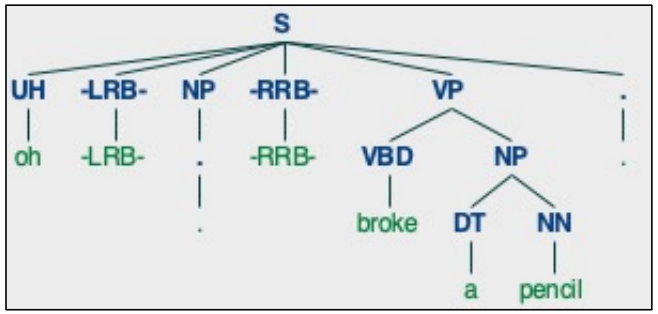
Childes Database

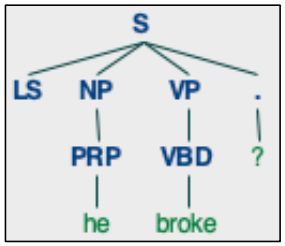


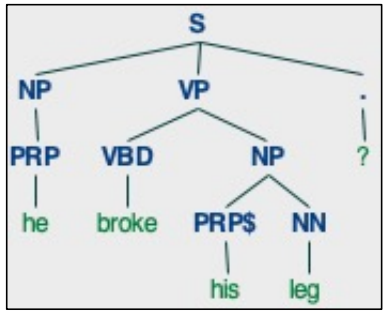
Childes Database

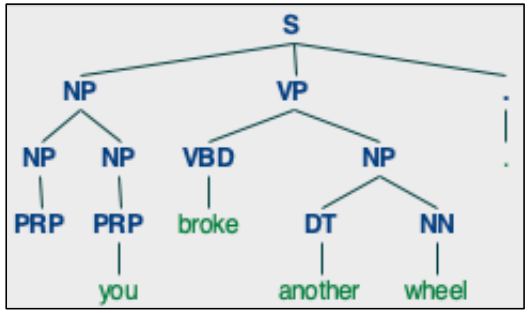












Childes Database: verb frames

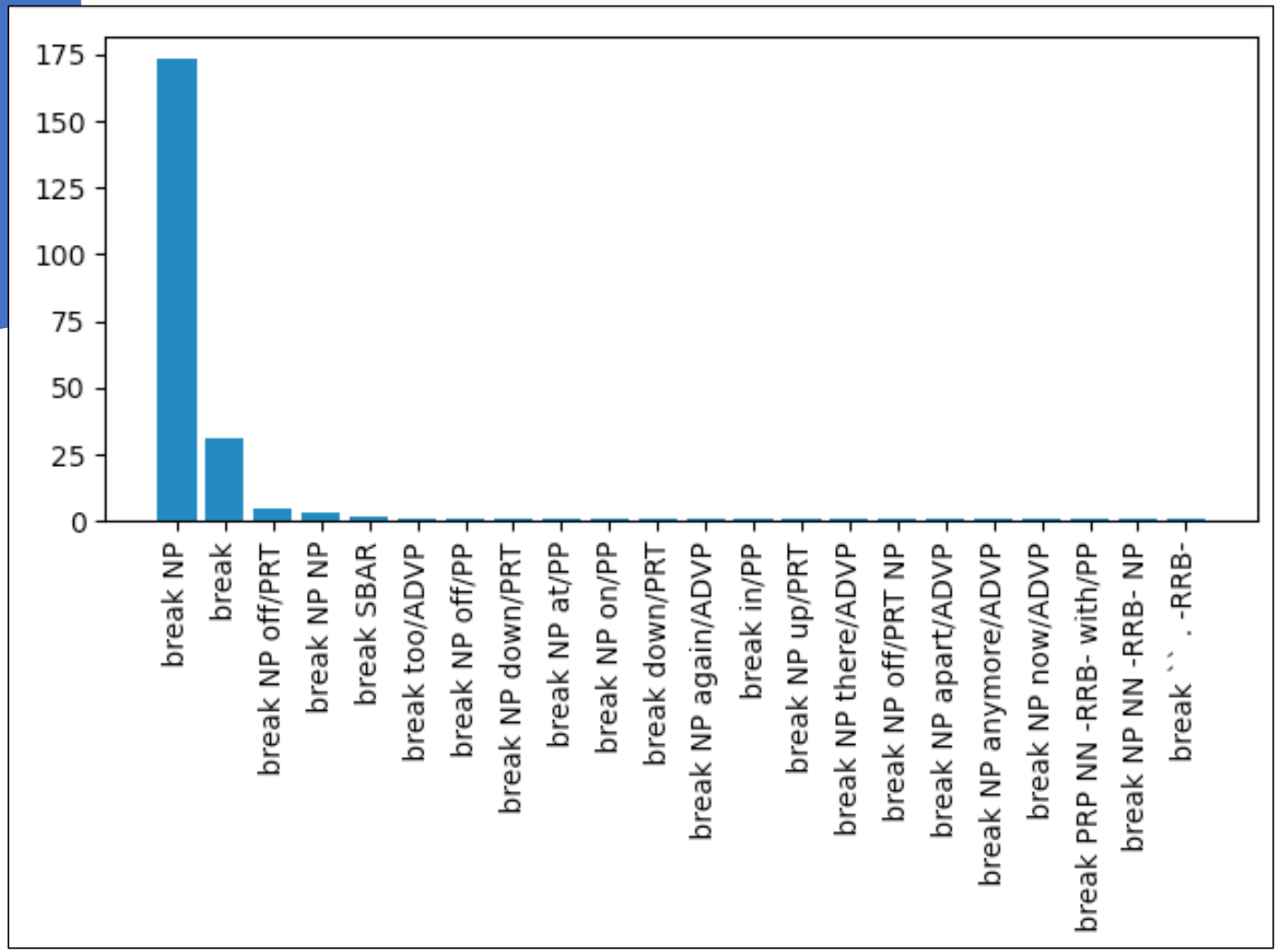
```
121# takes list of strings
122def sparsed_frames(l):
123    frames = []
124    for s in l:
125        sexpr = sparse(s)
126        if sexpr != None:
127            for st in Tree.fromstring(sexpr).subtrees():
128                found = vp_break(st)
129                if found:
130                    frames.append(found)
131    return frames
```

```
[' break through/ADVP SBAR-LOC', ' break NP', ' break with/PP SBAR-PRP', ' break
down/PRT into/PP', ' break up/PRT', ' break NP', ' break up/PRT NP', ' break NP', '
break', ' break , SBAR-ADV', ' break out/PRT', ' break (IN into)/PP', ' break up/PRT
NP', ' break NP for/PP', ' break NP', ' break out/PRT NP', ' break out/PRT *T*-
1/ADVP-TMP', ' break away/ADVP from/PP', ' break S', ' break into/PP', ' break
out/PP *T*-3/ADVP', ' break NP on/PP', ' break NP', ' break out/PP', ' break NP', '
break up/PRT into/PP , with/PP', ' VBP CC break NP at/PP-LOC', …, ' break for/PP-TMP
SBAR-ADV', ' break off/PRT NP abruptly/ADVP-MNR in/PP-TMP , SBAR-PRP']
```

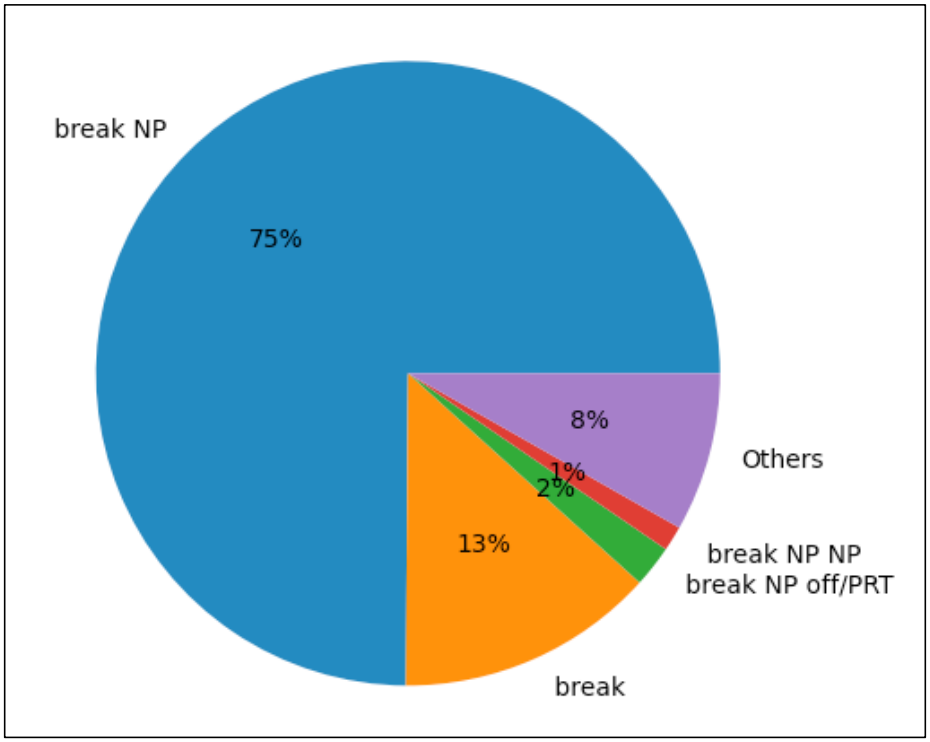
Childes Database: frame distribution

```
>>> frames2 = sparsed_frames(sents)
>>> fd = nltk.FreqDist(frames2)
>>> fd
FreqDist({' break NP': 173, ' break': 31, ' break NP off/PRT': 5, ' break NP NP': 3, ' break
SBAR': 2, ' break too/ADVP': 1, ' break NP off/PP': 1, ' break NP down/PRT': 1, ' break NP
at/PP': 1, ' break NP on/PP': 1, ...})
>>> fd.N()
231
>>> fd.B()
22
>>> fd.hapaxes()
[' break too/ADVP', ' break NP off/PP', ' break NP down/PRT', ' break NP at/PP', ' break NP
on/PP', ' break down/PRT', ' break NP again/ADVP', ' break in/PP', ' break NP up/PRT', '
break NP there/ADVP', ' break NP off/PRT NP', ' break NP apart/ADVP', ' break NP
anymore/ADVP', ' break NP now/ADVP', ' break PRP NN -RRB- with/PP', ' break NP NN -RRB- NP',
' break `` . -RRB-']
>>> len(fd.hapaxes())
17
```

Childes: *break* verb frame distribution



Childes Database: frame distribution



Matplotlib pie chart

- Verb frame distribution:

```
>>> len(sents)
238
>>> frames2 = sparsed_frames(sents)
>>> fd2 = nltk.FreqDist(frames2)
>>> fd2
FreqDist({' break NP': 173, ' break': 31, ' break NP off/PRT': 5, ' break NP NP': 3, ' break SBAR': 2, ' break too/ADVP': 1, ' break NP off/PP': 1, ' break NP down/PRT': 1, ' break NP at/PP': 1, ' break NP on/PP': 1, ...})
>>> fd2.N()
231
```

Matplotlib pie chart

- Create abbreviated pie chart:

1% or greater only

```
>>> fd3 = nltk.FreqDist({t[0]:t[1] for t in fd2.most_common() if t[1] > int(fd2.N() / 100)})
>>> fd3['others']  = fd2.N() - fd3.N()
>>> fd3
FreqDist({' break NP': 173, ' break': 31, 'others': 19, ' break NP off/PRT': 5, ' break NP NP': 3})
>>> plt.pie([t[1] for t in fd3.most_common()],labels=[t[0] for t in fd3.most_common()], autopct='%1.0f%%')
>>> plt.show()
```

matplotlib.axes.Axes.pie

```
Axes.pie(x, explode=None, labels=None, colors=None, autopct=None,
pctdistance=0.6, shadow=False, labeldistance=1.1, startangle=0,
radius=1, counterclock=True, wedgeprops=None, textprops=None,
center=(0, 0), frame=False, rotatelabels=False, *, normalize=True,
hatch=None, data=None)                                          [source]
```

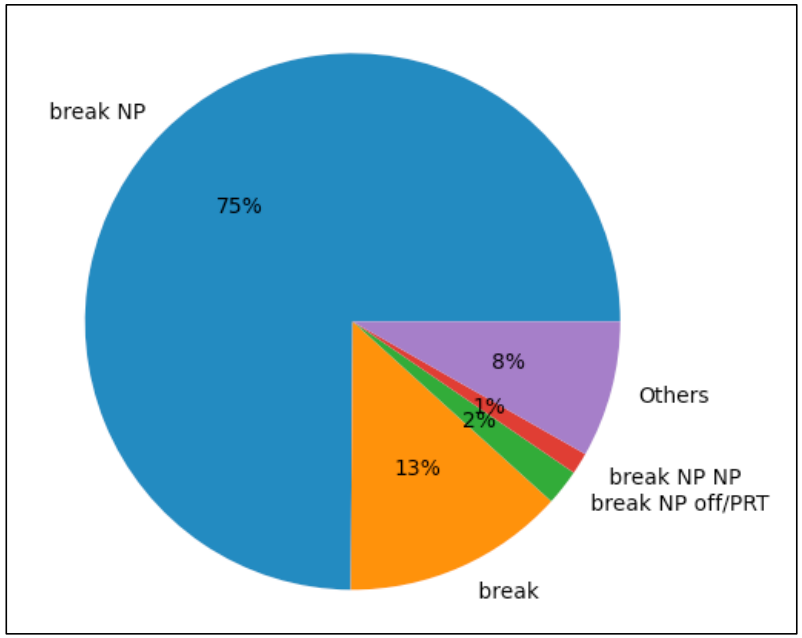
Plot a pie chart.

Make a pie chart of array *x*. The fractional area of each wedge is given by `x/sum(x)`.

The wedges are plotted counterclockwise, by default starting from the x-axis.

Break verb frames

Childes database



ptb