Department of
Linguistics and Translation
香港城市大學
City University of Hong Kong

Computational Linguistics
LT3233

Jixing Li

Lecture 2: POS tagging

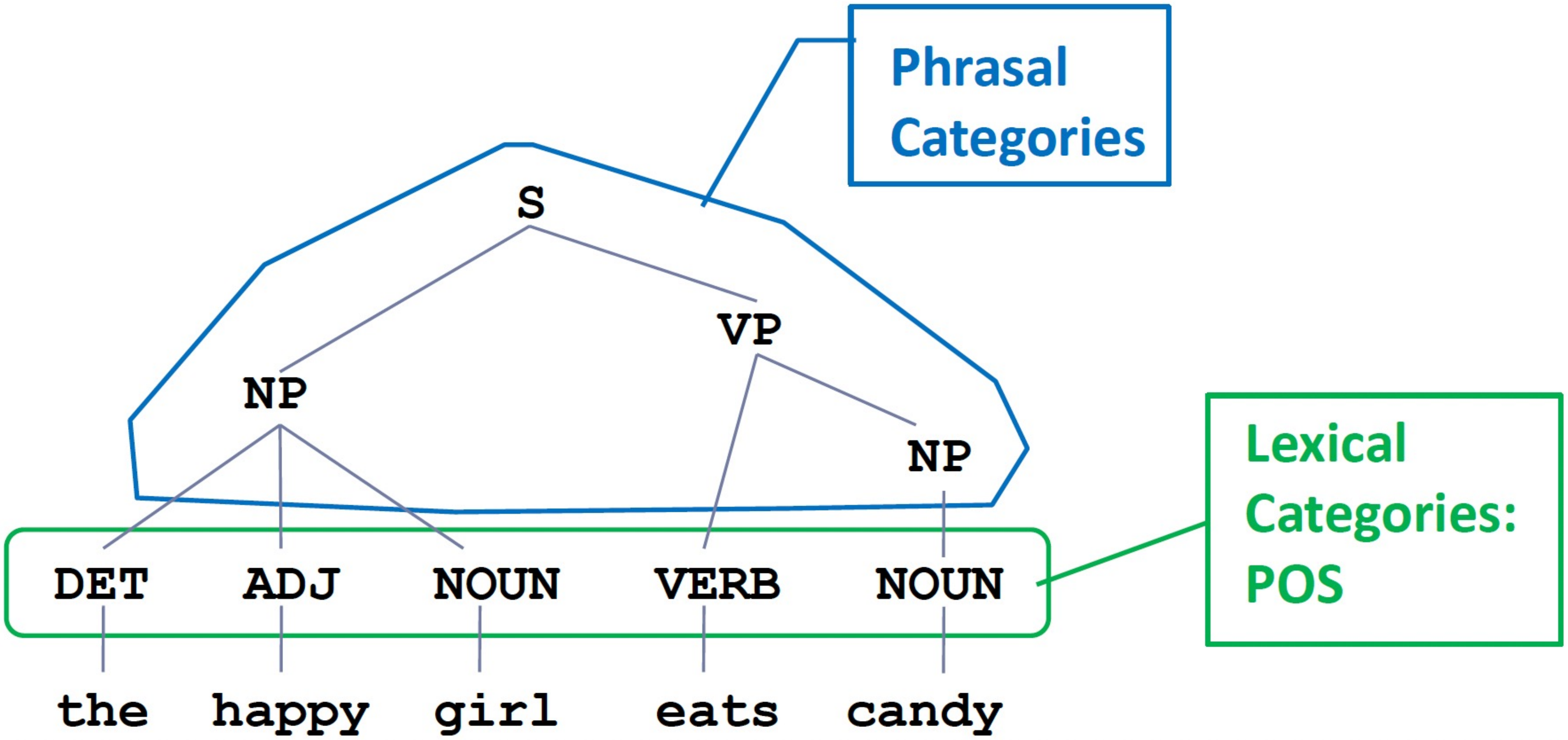
Slides adapted from John Hale

Lecture plan

- What are parts of speech (POS)?
- How to build a POS tagger?
- Short break (15 mins)
- Hands-on exercises

Phrasal vs lexical categories

A tree structure for “The happy girl eats candy”:



Part-of-speech

NOUN, **VERB**, **ADJ** (Adjective), **ADV** (Adverb)

→Open class words: words have semantic content

→New nouns and verbs are continually created: *iPhone*, *tweet, instagrammable, internet troll, etc.*

DET (Determiner), **ADP** (Adposition/Preposition), **PRON** (Pronoun), **PRT** (Particle), **CONJ** (Conjunction)

→Closed class words: grammatical/functional words

→Relatively fixed membership

Is that all?

- **Interjections**
  - *Uh oh*, that's too bad.
  - *Yes*, I'd like that.
  - That's nice, *eh*?
  - *Hooray*, she won gold.

- There are a few odd ones that are hard to classify:
  - **to** in infinitives:
    - I tried *to* finish. I went *to* school.
  - negative particle **not**
    - She did *not* eat. She is *not* happy.

- And many more...

POS in different datasets

tagset: A list of possible POS tags.

```
# Brown
nltk.corpus.brown.tagged_words()[:5] # f
```

```
[('The', 'AT'),
 ('Fulton', 'NP-TL'),
 ('County', 'NN-TL'),
 ('Grand', 'JJ-TL'),
 ('Jury', 'NN-TL')]
```

```
# Penn Treebank
nltk.corpus.treebank.tagged_words()[:5]
```

```
[('Pierre', 'NNP'),
 ('Vinken', 'NNP'),
 (',', ','),
 ('61', 'CD'),
 ('years', 'NNS')]
```

The Brown Corpus and the Penn Treebank Corpus

Brown: text samples of American English, of varied genres.
Penn Treebank: one million words of 1989 Wall Street Journal material annotated in a syntactic tree style.

Brown and the Penn Treebank

Brown:

The/at Fulton/np-tl County/nn-tl Grand/jj-tl Jury/nn-tl said/vbd Friday/nr an/at investigation/nn of/in Atlanta's/np$ recent/jj primary/nn election/nn produced/vbd ``/`` no/at evidence/nn "/" that/cs any/dti irregularities/nns took/vbd place/nn ./.

The/at jury/nn further/rbr said/vbd in/in term-end/nn presentments/nns that/cs the/at City/nn-tl Executive/jj-tl Committee/nn-tl ,/, which/wdt had/hvd over-all/jj charge/nn of/in the/at election/nn ,/, ``/`` deserves/vbz the/at praise/nn and/cc thanks/nns of/in the/at City/nn-tl of/in-tl Atlanta/np-tl "/" for/in the/at manner/nn in/in which/wdt the/at election/nn was/bedz conducted/vbn ./.

Penn Treebank

```
( (CODE (SYM SpeakerB1) (. .) ))
( (SBARQ
    (INTJ (UH So) )
    (WHNP-1
      (WHADJP (WRB how) (JJ many) )
      (, ,)
      (INTJ (UH um) )
      (, ,) (NN credit) (NNS cards) )
    (SQ (VBP do)
      (NP-SBJ (PRP you) )
      (VP (VB have)
        (NP (-NONE- *T*-1) )))
    (. ?) (-DFL- E_S) ))
( (CODE (SYM SpeakerA2) (. .) ))
( (S
    (INTJ (UH Um) )
    (, ,)
    (NP-SBJ (PRP I) )
    (VP (VBP think)
      (SBAR (-NONE- 0)
        (S
          (NP-SBJ (PRP I) )
          (VP (VBP 'm)
            (PP-PRD (IN down)
              (PP (IN to)
                (NP (CD one) )))))))
    (. .) (-DFL- E_S) ))
( (CODE (SYM SpeakerB3) (. .) ))
( (INTJ
    (INTJ (UH Oh) )
    (, ,)
    (INTJ (PRP$ my) (UH gosh) )
    (, ,) (-DFL- E_S) ))
( (S
    (NP-SBJ (PRP I) )
    (VP (VBP wish)
      (SBAR (-NONE- 0)
        (S
          (NP-SBJ (PRP I) )
          (VP (VBD was)
            (NP-MNR-PRD (DT that) (NN way) )))))
    (. .) (-DFL- E_S) ))
```

A universal POS tagset

The ‘**Universal Dependency**’ project (Nivre et al., 2016).

Tag	Meaning	English Examples
ADJ	adjective	*new, good, high, special, big, local*
ADP	adposition	*on, of, at, with, by, into, under*
ADV	adverb	*really, already, still, early, now*
CONJ	conjunction	*and, or, but, if, while, although*
DET	determiner, article	*the, a, some, most, every, no, which*
NOUN	noun	*year, home, costs, time, Africa*
NUM	numeral	*twenty-four, fourth, 1991, 14:24*
PRT	particle	*at, on, out, over per, that, up, with*
PRON	pronoun	*he, their, her, its, my, I, us*
VERB	verb	*is, say, told, given, playing, would*
.	punctuation marks	*. , ; !*
X	other	*ersatz, esprit, dunno, gr8, univeristy*

Change to universal tagset

```
# Universal tagset
nltk.corpus.brown.tagged_words(tagset='universal')[:5]
```

```
[('The', 'DET'),
 ('Fulton', 'NOUN'),
 ('County', 'NOUN'),
 ('Grand', 'ADJ'),
 ('Jury', 'NOUN')]
```

```
nltk.corpus.treebank.tagged_words(tagset='universal')[:5]
```

```
[('Pierre', 'NOUN'),
 ('Vinken', 'NOUN'),
 (',', '.'),
 ('61', 'NUM'),
 ('years', 'NOUN')]
```

POS ambiguity

- ~85% of words always have the same POS:
  - *she/***PRON**, *very/***ADV**
- ~15% of words can take on multiple POS:
  - Tim likes to go for *walks/***NOUN**. Joe *walks/***VERB** to school every day.
  - January is a *cold/***ADJ** month. I have a very bad *cold/***NOUN**.
  - I like *that/***DET** pie. I like *that/***NOUN**. I told you *that/***SCONJ** he's lying.
  -
- These ambiguous words tend to be very common.
- In *Brown Corpus*, 11.5% of all **word types** and 40% of **word tokens** are ambiguous!

Part-of-speech tagging

- POS tagging is a set of computer processes by which a single POS tag is assigned to each word, symbol, punctuations in a sentence.

The	happy	girl	eats	candy	.
DET	ADJ	NOUN	VERB	NOUN	.

- This is one of the earlier steps in an NLP task, following tokenization.

POS-tagged corpora in NLTK

- NLTK data include many corpus resources with POS tags.
  - The Brown Corpus, The Penn Treebank Corpus, NPS Chat Corpus, Chinese, Hindi, Spanish, Portuguese…
  - You can also load POS-tagged *words* or *sentences.*

```
nltk.corpus.treebank.tagged_words(tagset='universal')[0]
```

```
('Pierre', 'NOUN')
```

```
nltk.corpus.treebank.tagged_sents(tagset='universal')[0]
```

```
[('Pierre', 'NOUN'),
 ('Vinken', 'NOUN'),
 (',', '.'),
 ('61', 'NUM'),
   ...
```

NLTK's POS tagger

```
sent = 'The happy girl eats candy.'
tokens = word_tokenize(sent)
nltk.pos_tag(tokens, tagset='universal')
```

```
[('The', 'DET'),
 ('happy', 'ADJ'),
 ('girl', 'NOUN'),
 ('eats', 'NOUN'),
 ('candy', 'VERB'),
 ('.', '.')]
```

How would you design a POS tagger?

1. Tag everything a **NOUN**.
    - Why? Because **NOUN** is the most common POS.

    * Problem? Poor coverage.

2. Consider the morphology.
    - words end in 'ly' (*really*, *happily*) → **ADV**
    - words end in *'ed'* (*wanted*, *liked*)→ **VERB**

    * Problem? *'fly'* end in 'ly' but is not an adverb. Not every word has an identifiable morphological marker.

3. Maintain a dictionary of word and its POS. For each word, simply look up its tag in the dictionary.

    * Problem? Ambiguity.

    '*He has a question*/**NOUN**', '*He questioned*/**VERB** *the results*.'

N-gram taggers

1. The dictionary lists the most common POS tag for a word.
   - 'question' → **NOUN** (more frequent than **VERB**)

```
data[('question','NOUN')]
```

240

```
data[('question','VERB')]
```

17

Unigram Tagger

2. Instead of just individual word, the dictionary lists the most common tag for the **preceding POS + the word**.
   - ‘would/**AUX** *question*' → **VERB**
   - ‘the/**DET** *question*’ → **NOUN**

Bigram Tagger

- Why stop at just ***one*** preceding POS? Consider ***two***.
  - ‘water/**NOUN** is/**AUX** cold’ → **ADJ**
  - ‘have/**VERB** a/**DET** cold' → **NOUN**

Trigram Tagger

The bigger the context the better?

- trigram tagger always better than bigram tagger? bigram tagger always better than unigram tagger?

```
# test the taggers on unseen sentences
test_sents = word_tokenize('The happy girl eats candy.')
unigram_tagger.tag(test_sents)
```

```
[('The', 'DET'),
 ('happy', 'ADJ'),
 ('girl', 'NOUN'),
 ('eats', 'VERB'),
 ('candy', 'NOUN'),
 ('.', '.')]
```

```
bigram_tagger.tag(test_sents)
```

```
[('The', 'DET'),
 ('happy', 'ADJ'),
 ('girl', 'NOUN'),
 ('eats', None),
 ('candy', None),
 ('.', None)]
```

```
trigram_tagger.tag(test_sents)
```

```
[('The', 'DET'),
 ('happy', 'ADJ'),
 ('girl', 'NOUN'),
 ('eats', None),
 ('candy', None),
 ('.', None)]
```

The larger the context, the more specific it gets, the chance of a particular context not found in the corpus data increases.

→ the **sparse data problem**.

Addressing sparse data problem

Combine n-gram taggers as stacked back-off models:

1. Look up **"POSn-2 POSn-1** *word***"** in the *trigram* tagger.
2. If it's not found, look up **"POSn-1** *word***"** in the *bigram* tagger.
3. If it's not found, look up *"word"* in the *unigram* tagger.
4. If it's not found (unknown word), use the *Default Tagger* where everything gets tagged **NOUN**.

Stacked n-gram tagger

```
# train the stacked n-gram tagger
t0 = nltk.DefaultTagger('NOUN')
t1 = nltk.UnigramTagger(train_sents, backoff=t0)
t2 = nltk.BigramTagger(train_sents, backoff=t1)
t3 = nltk.TrigramTagger(train_sents, backoff=t2)
```

```
# test the stacked n-gram tagger
t3.tag(test_sents)
```

```
[('The', 'DET'),
 ('happy', 'ADJ'),
 ('girl', 'NOUN'),
 ('eats', 'VERB'),
 ('candy', 'NOUN'),
 ('.', '.')]
```

'*train*' and '*test*'?

Training and testing data

- When you build an NLP model using corpus data, you want to be able to **evaluate** it to see how well it performs.
  - But typically, you want to evaluate the performance on *unseen* data to make sure your model generalizes well to new sentences.
  - These unseen data should also have correct annotations, if you were to perform *automated* evaluation.
- Therefore, it is customary to partition your data into **two sets**:

Training data
(building model)

Testing data
(evaluating model)

Preparing training/testing datasets

```
brown_sents = nltk.corpus.brown.tagged_sents(tagset='universal')
```

```
len(brown_sents)
```

```
57340
```

```
size = round(len(brown_sents)*0.9)
```

```
size
```

```
51606
```

```
train_sents = brown_sents[:size]
test_sents = brown_sents[size:]
```

```
len(train_sents)
```

```
51606
```

```
len(test_sents)
```

```
5734
```

- Training data: first *90%* of the Brown Corpus
- Testing data: last *10%* of the same

Tokenized sentences are used for training, not tokenized words

Evaluating a tagger

Compare the output of a tagger with a human-labelled (presumed "correct") **gold standard**

```
t0.accuracy(test_sents)
```

```
0.18536912470400 87
```

```
t1.accuracy(test_sents)
```

```
0.9523899331531192
```

```
t2.accuracy(test_sents)
```

```
0.964355315270007
```

```
t3.accuracy(test_sents)
```

```
0.9663984409379518
```

Find the mistakes

```
guess = [(word,hypothesis) for s in test_sents for (word,hypothesis) in t3.tag(untag(s))]
```

```
wrong = [(word,hypothesis,actual,s) for ((word,hypothesis),(_,actual,s)) in zip (guess, [(w,t,s)
         for s in test_sents for (w,t) in s]) if hypothesis != actual and hypothesis is not None]
```

```
print(wrong[0])
```

```
('about', 'ADP', 'ADV', [('He', 'PRON'), ('was', 'VERB'), ('about', 'ADV'), ('50', 'NUM'), ('years', 'NOUN'),
```

Hard for humans too!

Evaluating a tagger

- But how good is "**good**"? 90%? 95%? 98%...?
  - We need to establish a baseline.
  - A good unigram tagger can already achieve **90-91% (!)**
  - Bigram/trigram … taggers should show a better performance.

- How about a ceiling?
  - Agreement between **human annotators** tops out at about 97%. Therefore, trained taggers cannot be expected to perform better than that.

To do

- **Leave a comment for Lecture 2!**

  https://jixing-li.github.io/comments.html
- Submit HW1
- Optional reading: **NTLK** Ch5; **SLP** Ch8.