Lecture 7: Sequence Tagging

PCL II, CL, UZH
April 13, 2016

Universität Zürich[UZH]

Contents

Tagging

Universität Zürich[UZH]

- Part-of-speech (PoS) tagging
  - Fruit flies like a banana
  - Time flies like an arrow

Tagging

Universität Zürich[UZH]

- Part-of-speech (PoS) tagging
  - Fruit$_{\text{noun}}$ flies$_{\textbf{noun}}$ like$_{\textbf{verb}}$ a$_{\text{determiner}}$ banana$_{\text{noun}}$
  - Time$_{\text{noun}}$ flies$_{\textbf{verb}}$ like$_{\textbf{preposition}}$ an$_{\text{determiner}}$ arrow$_{\text{noun}}$

Tagging

- Part-of-speech (PoS) tagging
  - Fruit$_{\text{noun}}$ flies$_{\textbf{noun}}$ like$_{\textbf{verb}}$ a$_{\text{determiner}}$ banana$_{\text{noun}}$
  - Time$_{\text{noun}}$ flies$_{\textbf{verb}}$ like$_{\textbf{preposition}}$ an$_{\text{determiner}}$ arrow$_{\text{noun}}$
- Named entity recognition
  - Prof. people gives a presentation in train
  - Prof. Volk hält einen Vortrag in Zug

Tagging

Universität Zürich[UZH]

- Part-of-speech (PoS) tagging
  - Fruit$_{\text{noun}}$ flies$_{\textbf{noun}}$ like$_{\textbf{verb}}$ a$_{\text{determiner}}$ banana$_{\text{noun}}$
  - Time$_{\text{noun}}$ flies$_{\textbf{verb}}$ like$_{\textbf{preposition}}$ an$_{\text{determiner}}$ arrow$_{\text{noun}}$
- Named entity recognition
  - Prof. ~~people~~Volk gives a presentation in ~~train~~Zug
  - Prof. Volk$_{\text{NAME}}$ hält einen Vortrag in Zug$_{\text{PLACE}}$

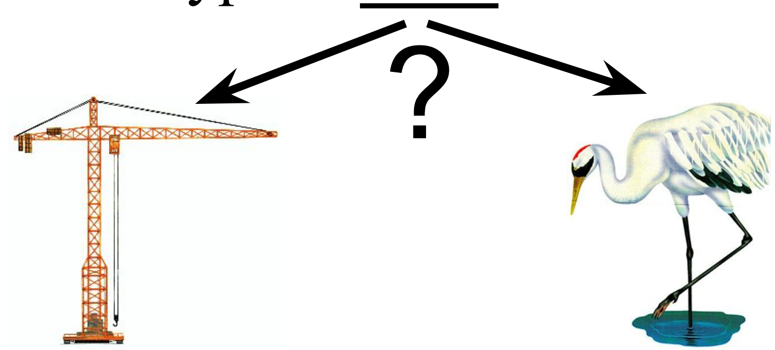
Tagging

Universität Zürich[UZH]

- Part-of-speech (PoS) tagging
  - Fruit$_{\text{noun}}$ flies$_{\textbf{noun}}$ like$_{\textbf{verb}}$ a$_{\text{determiner}}$ banana$_{\text{noun}}$
  - Time$_{\text{noun}}$ flies$_{\textbf{verb}}$ like$_{\textbf{preposition}}$ an$_{\text{determiner}}$ arrow$_{\text{noun}}$
- Named entity recognition
  - Prof. ~~people~~Volk gives a presentation in ~~train~~Zug
  - Prof. Volk$_{\text{NAME}}$ hält einen Vortrag in Zug$_{\text{PLACE}}$
- Word sense disambiguation
  - A duck is smaller than a typical crane

Tagging

- Part-of-speech (POS) tagging
  - Fruit$_{\text{noun}}$ flies$_{\textbf{noun}}$ like$_{\textbf{verb}}$ a$_{\text{determiner}}$ banana$_{\text{noun}}$
  - Time$_{\text{noun}}$ flies$_{\textbf{verb}}$ like$_{\textbf{preposition}}$ an$_{\text{determiner}}$ arrow$_{\text{noun}}$
- Named entity recognition
  - Prof. ~~people~~Volk gives a presentation in ~~train~~Zug
  - Prof. Volk$_{\text{NAME}}$ hält einen Vortrag in Zug$_{\text{PLACE}}$
- Word sense disambiguation
  - A duck is smaller than a typical crane



Tagging
POS Tagging

Universität Zürich[UZH]

- Part-of-speech (POS) tagging
  - Fruit$_{\text{noun}}$ flies$_{\textbf{noun}}$ like$_{\textbf{verb}}$ a$_{\text{determiner}}$ banana$_{\text{noun}}$
  - Time$_{\text{noun}}$ flies$_{\textbf{verb}}$ like$_{\textbf{preposition}}$ an$_{\text{determiner}}$ arrow$_{\text{noun}}$
- Named entity recognition
  - Prof. ~~people~~Volk gives a presentation in ~~train~~Zug
  - Prof. Volk$_{\text{NAME}}$ hält einen Vortrag in Zug$_{\text{PLACE}}$
- Word sense disambiguation
  - A duck is smaller than a typical crane

Contents

POS Tagging Applications

- Information extraction
- Question answering
- Speech synthesis
  - conTENT vs CONtent, obJECT vs OBject
- Parsing
- Word sense disambiguation
- Machine translation

POS Tagging
Taggers

- Tree-tagger
- Stanford POS tagger
- Zmorge
- …

POS Tagging
Taggers

```
:kitt$ echo "Fed raises interest rates 0.5 percent" |  tree-tagger-english
    reading parameters ...
    tagging ...
Fed            NP   Fed
raises         VVZ  raise
interest       NN   interest
rates          NNS  rate
0.5            CD   @card@
percent        NN   percent
     finished.
```

POS Tagging
Taggers

Use external tagger in Python:

```
import os

def tag(input):
     output = []
     taggerProc = os.popen("echo %s | tree-tagger-english"  %input)

     for line in taggerProc.readlines():
          (wordform, tag, lemma) = line.split("\t")

          lemma = lemma.strip()
          if lemma == "<unknown>":
               lemma = wordform

          output.append((wordform, tag, lemma))

     return output

if __name__ == "__main__":
     print tag("Fed raises interest rates 0.5 percent." )
```

POS Tagging
Taggers

NLTK built-in:

```
import nltk

tokenList = nltk.word_tokenize( "Fed raises interest rates 0.5 percent" )
# ['Fed', 'raises', 'interest', 'rates', '0.5', 'percent']

posResult = nltk.pos_tag(tokenList)
# [('Fed', 'NNP'), ('raises', 'VBZ'), ('interest', 'NN'), ('rates', 'NNS'),
('0.5', 'CD'), ('percent', 'NN')]
# posResult[0] = ('Fed', 'NNP')
# posResult[0][1] = 'NNP'
```

POS Tagging
Tagged Data

- The Brown corpus:
  - `nltk.corpus.`**`brown`**`.raw()`
  - `nltk.corpus.`**`brown`**`.words()`
  - `nltk.corpus.`**`brown`**`.sents()`
  - `nltk.corpus.`**`brown`**`.tagged_words()`

    `[(u'The', u'AT'), (u'Fulton', u'NP-TL'), …]`
  - `nltk.corpus.`**`brown`**`.tagged_sents()`

- also **`treebank`** (Penn), **`conll2007`**, etc.

POS Tagging
Tagged Data

Universität Zürich[UZH]

- The Brown corpus:
  - `nltk.corpus.`**`brown`**`.raw()`
  - `nltk.corpus.`**`brown`**`.words()`
  - `nltk.corpus.`**`brown`**`.sents()`
  - `nltk.corpus.`**`brown`**`.tagged_words()`

    `[(u’The’, u’AT’), (u’Fulton’, u’NP-TL’), …]`
  - `nltk.corpus.`**`brown`**`.tagged_sents()`

- also **`treebank`** (Penn), **`conll2007`**, etc.

→ Problem: different tag sets
  - NLTK solution: simplified universal tag set

POS Tagging
NLTK simplified tag set

Universität Zürich[UZH]

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*

POS Tagging
NLTK simplified tag set

```
>>> import nltk.corpus


>>> print nltk.corpus.brown.tagged_words()
# [('The', 'AT'), ('Fulton', 'NP-TL'), ...]

>>> print nltk.corpus.brown.tagged_words(tagset='universal')
# [('The', 'DET'), ('Fulton', 'NOUN'), ...]


>>> print nltk.corpus.treebank.tagged_words()
# [('Pierre', 'NNP'), ('Vinken', 'NNP'), ...]

>>> print nltk.corpus.treebank.tagged_words(tagset= 'universal')
# [('Pierre', 'NOUN'), ('Vinken', 'NOUN'), ...]
```

POS Tagging
How? - Introduction

Universität Zürich[UZH]

There are essentially two sources of information:

1. Syntagmatic Information

   Look at the tags of other words in the context

   e.g. a new *play*. NN or VBP?

   AT JJ NN vs. AT JJ VBP

2. Lexical Information

   Consider only the word involved

   → assign the most common tag.

Contents

Markov Model Taggers
Bigram tagger

Universität Zürich[UZH]

- Statistical: we need a tagged training corpus
- Maximum Likelihood Estimate

  $P(t^k \mid t^j) = C(t^k \mid t^j) / C(t^j)$ For tags

  $P(w^l \mid t^j) = C(w^l : t^j) / C(t^j)$ For words

  $t = P(w|\ t') \cdot P(t'|t_{i-1})$

  e.g. a new play

  P(NN|JJ) >> P(VBP|JJ)

  P(NN|JJ) ≈ 0.45 and P(VBP|JJ) ≈ 0.0005

Markov Model Taggers
Bigram tagger

- Problem:

  clearly marked is ambiguous in a Bigram Markov Model.

  RB (adverb) can precede both:

  → *VBD* (verb in the past tense)

  → *VBN* (past participle)

Markov Model Taggers
Bigram tagger

- Problem:
  clearly marked is ambiguous in a Bigram Markov Model.
  RB (adverb) can precede both:
  → *VBD* (verb in the past tense)
  → *VBN* (past participle)

- More context can help: Trigram tagger
  P(BEZ RB VBN|*is clearly marked*) > P(BEZ RB VBD|*is* ...)
  P(PN RB VBD|*he clearly marked*) > P(PN RB VBN|*he* …)

Markov Model Taggers
Bigram tagger

```
import nltk
from nltk.tag import *


bigram_tagger = nltk.BigramTagger(train_sents)
print bigram_tagger.tag(untag(sents[2007]))
print bigram_tagger.tag(untag(sents[4203]))
# [('The', 'AT'), ('population', 'NN'), ('of', 'IN'), ('the', 'AT'),
('Congo', 'NP'), ('is', 'BEZ'), ('13.5', None), ('million', None), (',',
None), ('divided', None), ('into', None), …
```

Markov Model Taggers
Bigram tagger

```
import nltk
from nltk.tag import *


bigram_tagger = nltk.BigramTagger(train_sents)
print bigram_tagger.tag(untag(sents[2007]))
print bigram_tagger.tag(untag(sents[4203]))
# [('The', 'AT'), ('population', 'NN'), ('of', 'IN'), ('the', 'AT'),
('Congo', 'NP'), ('is', 'BEZ'), ('13.5', None), ('million', None), (',',
None), ('divided', None), ('into', None), …
```

- problem: `'13.5'` is OOV

Markov Model Taggers
Backing off

- longer context (bigger *k*) means more probability that a particular n-gram has not been seen = “sparse data” effect

Markov Model Taggers
Backing off

Universität Zürich[UZH]

- longer context (bigger *k*) means more probability that a particular n-gram has not been seen = “sparse data” effect
- solution: back-off to smaller *k*:
  - try to find $p(t_i \mid t_{i-1}, t_{i-2})$
  - if "$t_{i-2}t_{i-1}t_i$" has not been seen, try $p(t_i \mid t_{i-1})$
  - etc.

Markov Model Taggers
Backing off

- Unigram Tagger: only based on $p(w_i|t_i)$
  - will assign the most probable tag per word
  - no matter the context

Markov Model Taggers
Backing off

- Unigram Tagger: only based on $p(w_i | t_i)$
  - will assign the most probable tag per word
  - no matter the context
- Default tagger: will assign the same tag to all words

Markov Model Taggers
Backing off

```
import nltk
from nltk.tag import *


default_tagger = nltk.DefaultTagger("NN")
unigram_tagger = nltk.UnigramTagger(train_sents,  backoff=default_tagger)
bigram_tagger = nltk.BigramTagger(train_sents,  backoff=unigram_tagger)


print bigram_tagger.tag(untag(sents[4203]))
# [('The', 'AT'), ('population', 'NN'), ('of', 'IN'), ('the', 'AT'),
('Congo', 'NP'), ('is', 'BEZ'), ('13.5', 'NN'), ('million', 'CD'), (',',
','), ('divided', 'VBN'), ('into', 'IN'),
```

Markov Model Taggers
My walk was awesome

$p(t \mid \text{"my"}) = \{\, \text{'pron'}: 0.99, \;\ldots \}$

$p(t \mid \text{"walk"}) = \{\, \text{'verb'}: 0.8,$
$\quad \text{'noun'}: 0.19, \;\ldots \}$

$p(t \mid \text{"was"}) = \{\, \text{'verb'}: 0.92, \;\ldots \}$

$p(t \mid \text{"awesome"}) = \{\, \text{'adj'}: 0.99, \;\ldots \}$

$p(t_i \mid t_{i-1} = \text{'<s>'}) = \{\, \text{'noun'}: 0.35, \; \text{'pron'}: 0.3 \;\ldots \}$

$p(t_i \mid t_{i-1} = \text{'pron'}) = \{\, \text{'verb'}: 0.3, \; \text{'noun'}: 0.35,$
$\quad \text{'adj'}: 0.3 \;\ldots \}$

$p(t_i \mid t_{i-1} = \text{'verb'}) = \{\, \text{'adj'}: 0.2, \; \text{'noun'}: 0.15,$
$\quad \text{'verb'}: 0.01, \;\ldots \}$

$p(t_i \mid t_{i-1} = \text{'noun'}) = \{\, \text{'verb'}: 0.3, \; \text{'noun'}: 0.2, \;\ldots \}$

Best tag for 'walk':
$p(\text{'verb'} \mid \text{'walk'}) \cdot p(\text{'verb'} \mid \text{'pron'}) = 0.8 \cdot 0.3 = 0.24$
$p(\text{'noun'} \mid \text{'walk'}) \cdot p(\text{'noun'} \mid \text{'pron'}) = 0.19 \cdot 0.35 = 0.0665$

Markov Model Taggers
My walk was awesome

Universität Zürich[UZH]

$p(t \mid \text{“my”}) = \{\ \text{‘pron’}:\ 0.99,\ \ldots\ \}$

$p(t \mid \text{“walk”}) = \{\ \text{‘verb’}:\ 0.8,$
$\quad \text{‘noun’}:\ 0.19,\ \ldots\ \}$

$p(t \mid \text{“was”}) = \{\ \text{‘verb’}:\ 0.92,\ \ldots\ \}$

$p(t \mid \text{“awesome”}) = \{\ \text{‘adj’}:\ 0.99,\ \ldots\ \}$

$p(t_i \mid t_{i-1} = \text{‘<s>’}) = \{\ \text{‘noun’}:\ 0.35,\ \text{‘pron’}:\ 0.3\ \ldots\ \}$

$p(t_i \mid t_{i-1} = \text{‘pron’}) = \{\ \text{‘verb’}:\ 0.3,\ \text{‘noun’}:\ 0.35,$
$\quad \text{‘adj’}:\ 0.3\ \ldots\ \}$

$p(t_i \mid t_{i-1} = \text{‘verb’}) = \{\ \text{‘adj’}:\ 0.2,\ \text{‘noun’}:\ 0.15,$
$\quad \text{‘verb’}:\ 0.01,\ \ldots\ \}$

$p(t_i \mid t_{i-1} = \text{‘noun’}) = \{\ \text{‘verb’}:\ 0.3,\ \text{‘noun’}:\ 0.2,\ \ldots\ \}$

Best tag for ‘walk’:

$p(\text{‘verb’} \mid \text{‘walk’}) \cdot p(\text{‘verb’} \mid \text{‘pron’}) = 0.8 \cdot 0.3 = 0.24$

$p(\text{‘noun’} \mid \text{‘walk’}) \cdot p(\text{‘noun’} \mid \text{‘pron’}) = 0.19 \cdot 0.35 = 0.0665$

But ‘walk’ as ‘verb’ brings down the likelihood of the whole sequence:

$p(\text{‘pron } \textbf{verb} \text{ verb adj’} \mid \text{‘my walk was awesome’}) =$

$\quad p(\text{‘pron’} \mid \text{‘my’}) \cdot \boldsymbol{p(\text{‘verb’} \mid \text{‘walk’})} \cdot p(\text{‘verb’} \mid \text{‘was’}) \cdot p(\text{‘adj’} \mid \text{‘awesome’}) \cdot$

$\quad p(\text{‘pron’} \mid \text{‘<s>’}) \cdot p(\text{‘verb’} \mid \text{‘pron’}) \cdot \boldsymbol{p(\text{‘verb’} \mid \text{‘verb’})} \cdot p(\text{‘adj’} \mid \text{‘verb’}) =$

$\quad 0.99 \cdot \mathbf{0.8} \cdot 0.92 \cdot 0.99 \cdot 0.3 \cdot 0.3 \cdot \mathbf{0.01} \cdot 0.2 = \mathbf{0.00013..}$

$p(\text{‘pron } \textbf{noun} \text{ verb adj’} \mid \text{‘my walk was awesome’}) =$

$\quad p(\text{‘pron’} \mid \text{‘my’}) \cdot \boldsymbol{p(\text{‘noun’} \mid \text{‘walk’})} \cdot p(\text{‘verb’} \mid \text{‘was’}) \cdot p(\text{‘adj’} \mid \text{‘awesome’}) \cdot$

$\quad p(\text{‘pron’} \mid \text{‘<s>’}) \cdot p(\text{‘verb’} \mid \text{‘pron’}) \cdot \boldsymbol{p(\text{‘noun’} \mid \text{‘verb’})} \cdot p(\text{‘adj’} \mid \text{‘verb’}) =$

$\quad 0.99 \cdot \mathbf{0.2} \cdot 0.92 \cdot 0.99 \cdot 0.3 \cdot 0.3 \cdot \mathbf{0.15} \cdot 0.2 = \mathbf{0.00049..}$

Viterbi Algorithm

Universität Zürich[UZH]

- Instead of
  $t = \text{argmax}_{t'}\ p(w_i|t') \cdot p(t'|t_{i-1})$ for each i

- Viterbi = optimization over the whole sequence:
  $\boldsymbol{t} = \text{argmax}_{\boldsymbol{t}'}\ p(\boldsymbol{t}'|\boldsymbol{w})$
- Using dynamic programming
  - to be explained in lecture #11

Hidden Markov Model

- No tagged training data

- In NLTK:

```
from nltk.tag.hmm import HiddenMarkovModelTagger
hmm_tagger = HiddenMarkovModelTagger.train(train_sents)
```

Contents

RegExp tagger

Universität Zürich[UZH]

- List of regular expressions and corresponding POS tags

```
>>> regexp_tagger = nltk.RegexpTagger(
...     [(r'^-?[0-9]+(.[0-9]+)?$', 'CD'),    # cardinal numbers
...      (r'(The|the|A|a|An|an)$', 'AT'),    # articles
...      (r'.*able$', 'JJ'),                 # adjectives
...      (r'.*ness$', 'NN'),                 # nouns formed from adjectives
...      (r'.*ly$', 'RB'),                   # adverbs
...      (r'.*s$', 'NNS'),                   # plural nouns
...      (r'.*ing$', 'VBG'),                 # gerunds
...      (r'.*ed$', 'VBD'),                  # past tense verbs
...      (r'.*', 'NN')                       # nouns (default)
... ])
>>> regexp_tagger.tag(test_sent)
[('The', 'AT'), ('Fulton', 'NN'), ('County', 'NN'), ('Grand', 'NN'), ('Jury', 'NN'),
('said', 'NN'), ('Friday', 'NN'), ('an', 'AT'), ('investigation', 'NN'), ('of', 'NN'),
("Atlanta's", 'NNS'), ('recent', 'NN'), ('primary', 'NN'), ('election', 'NN'),
('produced', 'VBD'), ('``', 'NN'), ('no', 'NN'), ('evidence', 'NN'), ("''", 'NN'),
('that', 'NN'), ('any', 'NN'), ('irregularities', 'NNS'), ('took', 'NN'),
('place', 'NN'), ('.', 'NN')]
```

Brill tagger

- start with a naive tagger's output
  - e.g. a unigram tagger:

    to/DT increase/NN grants/NN …

- The learning algorithm constructs a ranked list of transformations

  e.g.

Source tag	Target tag	Triggering environment
NN	VB	previous tag is TO
VBP	VB	one of the prev. 3 tags is MD
VBP	VB	one of the prev. 2 words is *n't*

Brill tagger

Universität Zürich[UZH]

```
C0 := corpus with each word tagset with its most frequent tag
for k := 0 step 1 do
     v := the transformation ui that minimizes E(ui(Ck))
     if (E(Ck) - E(v(Ck))) < e then break fi
     Ck+1 := v(Ck)
     Tk+1 := v
end
Output sequence: T1,...,Tk
```

C_0 := corpus with each word tagset with its most frequent tag

for k := 0 step 1 do

  v := the transformation u_i that minimizes $E(u_i(C_k))$

  if $(E(C_k) - E(v(C_k))) < e$ then break fi

  C_{k+1} := $v(C_k)$

  T_{k+1} := v

end

Output sequence: $T_1,...,T_k$

Contents

Accuracy in NLTK

```
import nltk
from nltk.tag import *


default_tagger = nltk.DefaultTagger("NN")
unigram_tagger = nltk.UnigramTagger(train_sents, backoff=default_tagger)
bigram_tagger = nltk.BigramTagger(train_sents, backoff=unigram_tagger)


print bigram_tagger.evaluate(test_sents)
```

Confusion matrix in NLTK

```
def tagList(sents):
    '''remove tokens and leave only tags'''
    return [tag for sent in sents for word, tag in sent]

def applyTagger(tagger, corpus):
    '''apply a tagger to a corpus'''
    return [tagger.tag(nltk.tag.untag(sent))  for sent in corpus]

goldTags = tagList(test_sents)
testTags = tagList(applyTagger(bigram_tagger, test_sents))

cm = nltk.ConfusionMatrix(goldTags, testTags)
print cm.pp(sort_by_count=True, show_percents=True,
    truncate=9)
```

Confusion matrix in NLTK

```
    |                           N                                   |
    |      N      I      A      N             J             N      C |
    |      N      N      T      S      ,      J      .      P      C |
----+-----------------------------------------------------------------+
 NN | <12.0%>     .   0.0%   0.1%      .   0.2%      .      .      . |
 IN |   0.0% <10.1%>     .      .      .      .      .      .      . |
 AT |      .      . <8.5%>      .      .      .      .      .      . |
NNS |   1.9%      .      . <4.3%>      .      .      .      .      . |
  , |      .      .      .      . <6.0%>      .      .      .      . |
 JJ |   1.4%      .      .      .      . <3.4%>      .   0.0%      . |
  . |      .      .      .      .      .      . <4.8%>      .      . |
 NP |   1.7%      .      .      .      .      .      . <2.7%>      . |
 CC |      .      .      .      .      .      .      .      . <2.8%>|
----+-----------------------------------------------------------------+
(row = reference; col = test)
```

Performance in practice

Universität Zürich[UZH]

Is 95% good accuracy?

Performance in practice

Universität Zürich[UZH]

Is 95% good accuracy?

Depends not just on the tagger

- The amount of training data available
- The tag set
- Same or different domain
- but on the text domain
- the language
- ...

State of the art:

http://aclweb.org/aclwiki/index.php?title=POS_Tagging_%28State_of_the_art%29

Lecture 7: Sequence Tagging

PCL II, CL, UZH
April 13, 2016

Universität Zürich[UZH]