Sequence Labeling

Phạm Quang Nhật Minh
Aimesoft JSC
minhpham0902@gmail.com

Lecture outline

- POS Tagging and Named Entity Recognition (NER)
- Hidden Markov Models for Part-of-Speech Tagging
- Conditional Random Fields
- Evaluation of Named Entity Recognition

Lecture outline

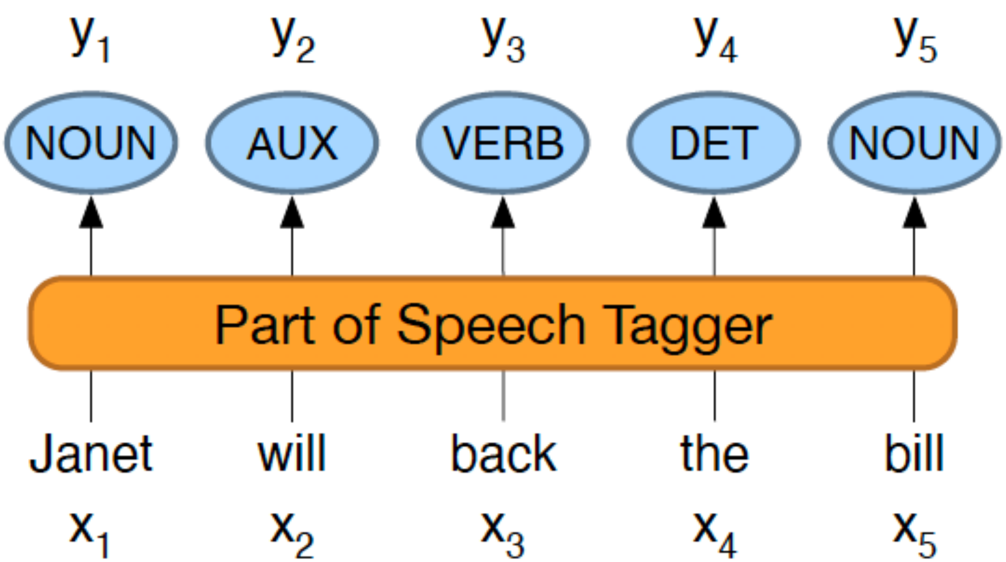
- Sequence Labeling Problems
- Hidden Markov Models for Part-of-Speech Tagging
- Conditional Random Fields
- Evaluation of Named Entity Recognition

Part-of-Speech Tagging

- Assigning a part-of-speech to each word in a text.
- Words often have more than one POS.
- **book**:
  - VERB: (***Book** that flight*)
  - NOUN: (*Hand me that **book***).

Part-of-Speech Tagging

- INPUT:
  - Jane will back the bill
- OUTPUT:
  - Jane/NOUN will/AUX back/VERB the/DET bill/NOUN



Why POS Tagging?

- Can be useful for other NLP tasks
  - Parsing: POS tagging can improve syntactic parsing
  - MT: reordering of adjectives and nouns (say from Spanish to English)
  - Sentiment or affective tasks: may want to distinguish adjectives or other POS
  - Text-to-speech (how do we pronounce "lead" or "object"?)

Challenges in POS tagging

- Words have more than one possible POS
  - book that flight
  - hand me that book
- Simple solution with dictionary look-up does not work in practice
  - One needs to determine the POS tag for an instance of a word from its context

Define a tagset

- We must agree on a standard inventory of word classes
  - Taggers are trained on a labeled corpora
  - The tagset needs to capture semantically or syntactically important distinctions that can easily be made by trained human annotators

Public tagsets in NLP

- Brown corpus - Francis and Kucera 1961
  - 87 tags
- 45-tag Penn Treebank tagset - - Marcus et al. 1993
  - Hand-annotated corpus of Wall Street Journal, 1M words
  - 45 tags, a simplified version of Brown tag set
  - Standard for English now
    - Most statistical POS taggers are trained on this Tagset

Penn Treebank tagset

Tag	Description	Example	Tag	Description	Example	Tag	Description	Example
CC	coordinating conjunction	*and, but, or*	PDT	predeterminer	*all, both*	VBP	verb non-3sg present	*eat*
CD	cardinal number	*one, two*	POS	possessive ending	*'s*	VBZ	verb 3sg pres	*eats*
DT	determiner	*a, the*	PRP	personal pronoun	*I, you, he*	WDT	wh-determ.	*which, that*
EX	existential 'there'	*there*	PRP$	possess. pronoun	*your, one's*	WP	wh-pronoun	*what, who*
FW	foreign word	*mea culpa*	RB	adverb	*quickly*	WP$	wh-possess.	*whose*
IN	preposition/ subordin-conj	*of, in, by*	RBR	comparative adverb	*faster*	WRB	wh-adverb	*how, where*
JJ	adjective	*yellow*	RBS	superlatv. adverb	*fastest*	$	dollar sign	$
JJR	comparative adj	*bigger*	RP	particle	*up, off*	#	pound sign	#
JJS	superlative adj	*wildest*	SYM	symbol	+,%, &	“	left quote	‘ or “
LS	list item marker	*1, 2, One*	TO	“to”	*to*	”	right quote	’ or ”
MD	modal	*can, should*	UH	interjection	*ah, oops*	(	left paren	[, (, {, <
NN	sing or mass noun	*llama*	VB	verb base form	*eat*	)	right paren	],), }, >
NNS	noun, plural	*llamas*	VBD	verb past tense	*ate*	,	comma	,
NNP	proper noun, sing.	*IBM*	VBG	verb gerund	*eating*	.	sent-end punc	. ! ?
NNPS	proper noun, plu.	*Carolinas*	VBN	verb past part.	*eaten*	:	sent-mid punc	: ; ... – -

Named Entities

- Named entity, in its core usage, means anything that can be referred to with a proper name. Most common 4 tags:
  - PER (Person): “Marie Curie”
  - LOC (Location): “New York City”
  - ORG (Organization): “Stanford University”
  - GPE (Geo-Political Entity): "Boulder, Colorado”
- Often multi-word phrases
- But the term is also extended to things that aren't entities:
  - dates, times, prices

Named Entity Tagging

- The task of named entity recognition (NER):
  - find spans of text that constitute proper names
  - tag the type of the entity.

NER Output

Citing high fuel prices, [$_{\text{ORG}}$ **United Airlines**] said [$_{\text{TIME}}$ **Friday**] it has increased fares by [$_{\text{MONEY}}$ **$6**] per round trip on flights to some cities also served by lower-cost carriers. [$_{\text{ORG}}$ **American Airlines**], a unit of [$_{\text{ORG}}$ **AMR Corp.**], immediately matched the move, spokesman [$_{\text{PER}}$ **Tim Wagner**] said. [$_{\text{ORG}}$ **United**], a unit of [$_{\text{ORG}}$ **UAL Corp.**], said the increase took effect [$_{\text{TIME}}$ **Thursday**] and applies to most routes where it competes against discount carriers, such as [$_{\text{LOC}}$ **Chicago**] to [$_{\text{LOC}}$ **Dallas**] and [$_{\text{LOC}}$ **Denver**] to [$_{\text{LOC}}$ **San Francisco**].

Why NER?

- Sentiment analysis: consumer's sentiment toward a particular company or person?
- Question Answering: answer questions about an entity?
- Information Extraction: Extracting facts about entities from text.

Why NER is hard

- Segmentation
  - In POS tagging, no segmentation problem since each word gets one tag.
  - In NER we have to find and segment the entities!
- Type ambiguity

[$_{\text{PER}}$ Washington] was born into slavery on the farm of James Burroughs.
[$_{\text{ORG}}$ Washington] went up 2 games to 1 in the four-game series.
Blair arrived in [$_{\text{LOC}}$ Washington] for what may well be his last state visit.
In June, [$_{\text{GPE}}$ Washington] passed a primary seatbelt law.

BIO Tagging

- Define many new tags
  - B-PERS, B-DATE,…: beginning of a mention of a person/date...
  - I-PERS, I-DATE,…: inside of a mention of a person/date...
  - O: outside of any mention of a named entity

```
[PERS Pierre Vinken] , 61 years old , will join
[ORG IBM] 's board as a nonexecutive director
[DATE Nov. 2] .
```

```
Pierre_B-PERS Vinken_I-PERS ,_O 61_O years_O old_O ,_O
will_O join_O IBM_B-ORG 's_O board_O as_O a_O
nonexecutive_O director_O Nov._B-DATE 29_I-DATE ._O
```

BIO Tagging variants: IO and BIOES

[PER Jane Villanueva] of [ORG United] , a unit of [ORG United Airlines Holding] , said the fare applies to the [LOC Chicago] route.

Words	IO Label	BIO Label	BIOES Label
Jane	I-PER	B-PER	B-PER
Villanueva	I-PER	I-PER	E-PER
of	O	O	O
United	I-ORG	B-ORG	B-ORG
Airlines	I-ORG	I-ORG	I-ORG
Holding	I-ORG	I-ORG	E-ORG
discussed	O	O	O
the	O	O	O
Chicago	I-LOC	B-LOC	S-LOC
route	O	O	O
.	O	O	O

Standard algorithms for NER

Supervised Machine Learning given a human-labeled training set of text annotated with tags

- Hidden Markov Models
- Conditional Random Fields (CRF)/ Maximum Entropy Markov Models (MEMM)
- Neural sequence models (RNNs or Transformers)
- Large Language Models (like BERT), finetuned

Word Segmentation as Sequence Labeling

Chiều 28/2 , Hà Nội đã tổ chức họp trực tuyến về việc phòng chống dịch Covid-19 do ông Nguyễn Đức Chung - Chủ tịch UBND TP Hà Nội chủ trì .

Chiều 28/2 , Hà_Nội đã tổ_chức họp trực_tuyến về việc phòng_chống dịch Covid-19 do ông Nguyễn_Đức_Chung - Chủ_tịch UBND TP Hà_Nội chủ_trì .

BI Tagging

Chiều 28/2 , Hà Nội đã tổ chức họp trực tuyến về việc phòng chống dịch Covid-19 do ông Nguyễn Đức Chung - Chủ tịch UBND TP Hà Nội chủ trì .

Chiều/B 28/2/B ,/B Hà/B Nội/I đã/B tổ/B chức/I họp/B trực/B tuyến/I về/B việc/B phòng/B chống/I dịch/B Covid-19/B do/B ông/B Nguyễn/B Đức/I Chung/I -/B Chủ/B tịch/I UBND/B TP/B Hà/B Nội/I chủ/B trì/I ./B

Lecture outline

- Sequence Labeling Problems
- Hidden Markov Models for Part-of-Speech Tagging
- Conditional Random Fields
- Evaluation of Named Entity Recognition

Sequence Labeling

- Sequence Labeling
  - Input: a word (token) sequence $x_1 \ldots x_n$
  - Output: a tag sequence $y_1 \ldots y_n$

- In the supervised setting, we have a list of training examples $(x^{(i)}, y^{(i)})$ for $i = 1 \ldots m$ where
  - $x^{(i)}$ is a sentence $x_1^{(i)} \ldots x_{n_i}^{(i)}$ and $y^{(i)}$ is a tag sequence $y_1^{(i)} \ldots y_{n_i}^{(i)}$
  - *We learn a mapping from a word sequence to a tag sequence*

Supervised Learning Problem

- Training set:

  1 Pierre/NNP Vinken/NNP ,/, 61/CD years/NNS old/JJ ,/, will/MD join/VB the/DT board/NN as/IN a/DT nonexecutive/JJ director/NN Nov./NNP 29/CD ./.

  2 Mr./NNP Vinken/NNP is/VBZ chairman/NN of/IN Elsevier/NNP N.V./NNP ,/, the/DT Dutch/NNP publishing/VBG group/NN ./.

  3 Rudolph/NNP Agnew/NNP ,/, 55/CD years/NNS old/JJ and/CC chairman/NN of/IN Consolidated/NNP Gold/NNP Fields/NNP PLC/NNP ,/, was/VBD named/VBN a/DT nonexecutive/JJ director/NN of/IN this/DT British/JJ industrial/JJ conglomerate/NN ./.

  ...

  38,219 It/PRP is/VBZ also/RB pulling/VBG 20/CD people/NNS out/IN of/IN Puerto/NNP Rico/NNP ,/, who/WP were/VBD helping/VBG Huricane/NNP Hugo/NNP victims/NNS ,/, and/CC sending/VBG them/PRP to/TO San/NNP Francisco/NNP instead/RB ./.

- From the training set, induce a function/algorithm that maps new sentences to their tag sequences.

Hidden Markov Models (HMM) for Tagging

- We have an input sentence $x = x_1, x_2, \ldots, x_n$
  - (x_i is the i'th word in the sentence)
- We have a tag sequence $y = y_1, y_2, \ldots, y_n$
  - (y_i is the i'th tag in the sentence)

- We'll use an HMM to define

$$p(x_1, x_2, \ldots, x_n, y_1, y_2, \ldots, y_n)$$

for any sentence $x_1 x_2 \ldots x_n$ and tag sequence $y_1 y_2 \ldots y_n$ of the same length.

HMM tagger

- The most likely tag sequence for x is

$$\arg\max_{y_1 \ldots y_n} p(x_1 \ldots x_n, y_1 \ldots y_n)$$
$$= \arg\max_{y_1 \ldots y_n} p(y_1 \ldots y_n) p(x_1 \ldots x_n | y_1 \ldots y_n)$$

- How can we decompose the equation into simpler terms?

Assumptions in first-order HMMs

- **Markov Assumption**: The probability of a hidden state depends only on its previous hidden state.

$$P(y_i|y_1 \dots y_{i-1}) = P(y_i|y_{i-1})$$

- **Observation Independence Assumption**: The probability of an observation depends only on its associated hidden state.

$$P(x_i|x_1 \dots x_i \dots x_n, y_1 \dots y_i \dots y_n) = P(x_i|y_i)$$

First-order (bigram) Hidden Markov Models

- For any sentence $x = x_1 \dots x_n$ where $x_i \in V$ for $x = i = 1 \dots n$, and any tag sequence $y = y_1 \dots y_n$ where $y_i \in S$ for $i = 1 \dots n$, and y_{n+1} = </s>, the joint probability of the sentence and tag sequence is

$$p(x_1 \dots x_n, y_1 \dots y_{n+1}) \approx \prod_{i=1}^{n+1} P_T(y_i | y_{i-1}) \prod_{i=1}^{n} P_E(x_i | y_i)$$

Transition probabilities   Emission probabilities

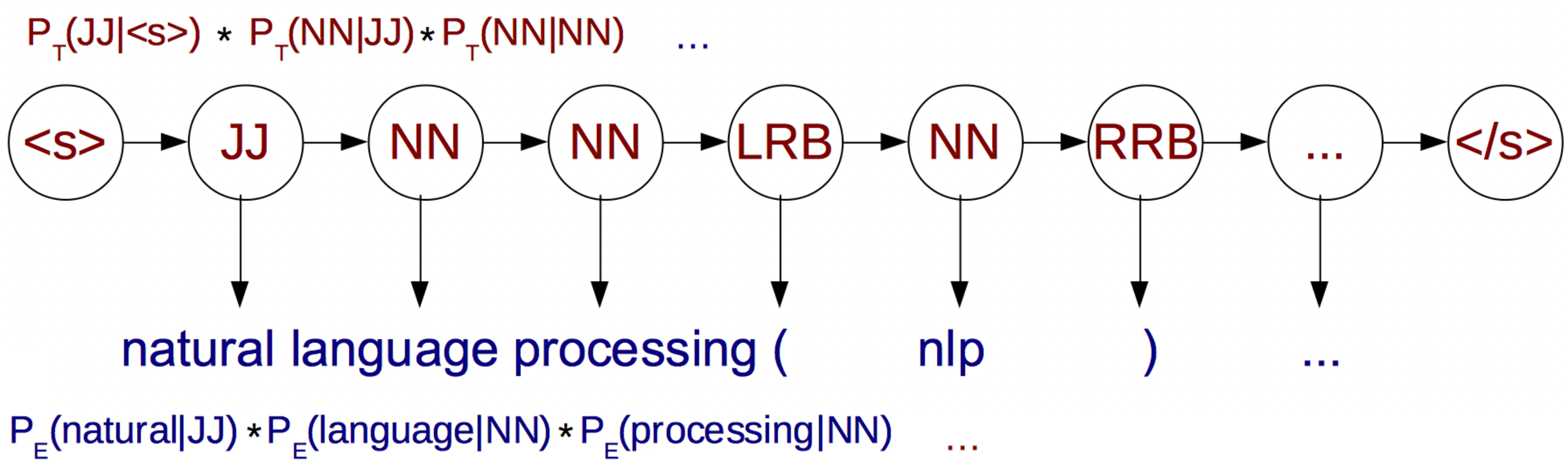
Hidden Markov Models for POS Tagging

- POS → POS transition probabilities

$$P(Y) \approx \prod_{i=1}^{n+1} P_T(y_i|y_{i-1})$$

- POS → Word emission probabilities

$$P(X|Y) \approx \prod_{i=1}^{n} P_E(x_i|y_i)$$



An Example

- If we have $n = 3$, $x_1 \ldots x_3$ equal to the sentence *the dog laughs*
  and $y_1 \ldots y_4$ equal to the tag sequence D N V </s>, then

$$P(x_1 \ldots x_n, y_1 \ldots y_{n+1})$$

$$\approx P_T(D|\text{<s>}) \times P_T(N|D) \times P_T(V|N) \times P(\text{</s>}|\mathrm{V}) \times$$

$$P_E(the|D) \times P_E(dog|N) \times P_E(laughs|V)$$

Hidden Markov Models

Definition

$Q = q_1 q_2 \ldots q_N$	a set of N **states**
$A = a_{11} \ldots a_{ij} \ldots a_{NN}$	a **transition probability matrix** A, each a_{ij} representing the probability of moving from state i to state j, s.t. $\sum_{j=1}^{N} a_{ij} = 1 \quad \forall i$
$O = o_1 o_2 \ldots o_T$	a sequence of T **observations**, each one drawn from a vocabulary $V = v_1, v_2, ..., v_V$
$B = b_i(o_t)$	a sequence of **observation likelihoods**, also called **emission probabilities**, each expressing the probability of an observation o_t being generated from a state q_i
$\pi = \pi_1, \pi_2, ..., \pi_N$	an **initial probability distribution** over states. π_i is the probability that the Markov chain will start in state i. Some states j may have $\pi_j = 0$, meaning that they cannot be initial states. Also, $\sum_{i=1}^{n} \pi_i = 1$

Learning Hidden Markov Models (with tags)

- Count the number of occurrences in the corpus and

natural language processing (nlp) is …

c(JJ→natural)++ ↑ ↑ c(NN→language)++ …

<s> → JJ → NN NN LRB NN RRB VB … </s>

c(<s> JJ)++ c(JJ NN)++ …

- Divide by context to get probability

$$P_T(\text{LRB}|\text{NN}) = c(\text{NN LRB})/c(\text{NN}) = 1/3$$

$$P_E(\text{language}|\text{NN}) = c(\text{NN} \rightarrow \text{language})/c(\text{NN}) = 1/3$$

Learning Hidden Markov Models (with tags)

- Transition probabilities

$$P_T(t_i|t_{i-1}) = \frac{C(t_{i-1}, t_i)}{C(t_{i-1})}$$

- Emission probabilities

$$P_E(w_i|t_i) = \frac{C(t_i, w_i)}{C(t_i)}$$

Note: Smoothing

- HMM transition probabilities: there are not many tags, so smoothing may not be necessary

$$P(t_i|t_{i-1}) = \lambda_1 \frac{C(t_{i-1},t_i)}{C(t_{i-1})} + (1\text{-}\ \lambda_1) \frac{C(t_i)}{C()}$$

- HMM emission probabilities: smooth for unknown words

$$P_E(w_i|t_i) = \lambda \frac{C(t_i,w_i)}{C(t_i)} + (1\text{-}\lambda) \frac{1}{N}$$

Training algorithm


```
# Input data format is "natural language …"
make a map emit, transition, context
for each line in file
    previous = "<s>"                          # Make the sentence start
    context[previous]++
    split line into wordtags with " "
    for each wordtag in wordtags
        split wordtag into word, tag with "_"
        transition[previous+" "+tag]++ # Count the transition
        context[tag]++                        # Count the context
        emit[tag+" "+word]++                  # Count the emission
        previous = tag
    transition[previous+" </s>"]++
# Print the transition probabilities
for each key, value in transition
    split key into previous, word with " "
    print "T", key, value/context[previous]
# Do the same thing for emission probabilities with "E"
```

HMM tagging as decoding

- Given as input HMM $\lambda = (A, B)$, and a sequence of observation $O = o_1, o_2, \dots, o_T$, find the most probable sequence of states $Q = q_1 q_2 \dots q_T$

HMM decoding in POS tagging

- Input: a sequence of n words $w_1 \dots w_n$
- Output: most probable tag sequence $t_1 \dots t_n$

$$\hat{t}_{1:n} = \underset{t_1 \dots t_n}{\operatorname{argmax}} P(t_1 \dots t_n | w_1 \dots w_n)$$

HMM decoding

$$\hat{t}_{1:n} = \underset{t_1 \ldots t_n}{\operatorname{argmax}} P(t_1 \ldots t_n | w_1 \ldots w_n)$$

Applying Bayesion Rule

$$\hat{t}_{1:n} = \underset{t_1 \ldots t_n}{\operatorname{argmax}} \frac{P(w_1 \ldots w_n | t_1 \ldots t_n) P(t_1 \ldots t_n)}{P(w_1 \ldots w_n)}$$

Dropping denominator

$$\hat{t}_{1:n} = \underset{t_1 \ldots t_n}{\operatorname{argmax}} P(w_1 | \ldots w_n | t_1 \ldots t_n) P(t_1 \ldots t_n)$$

HMM decoding

Observation independence

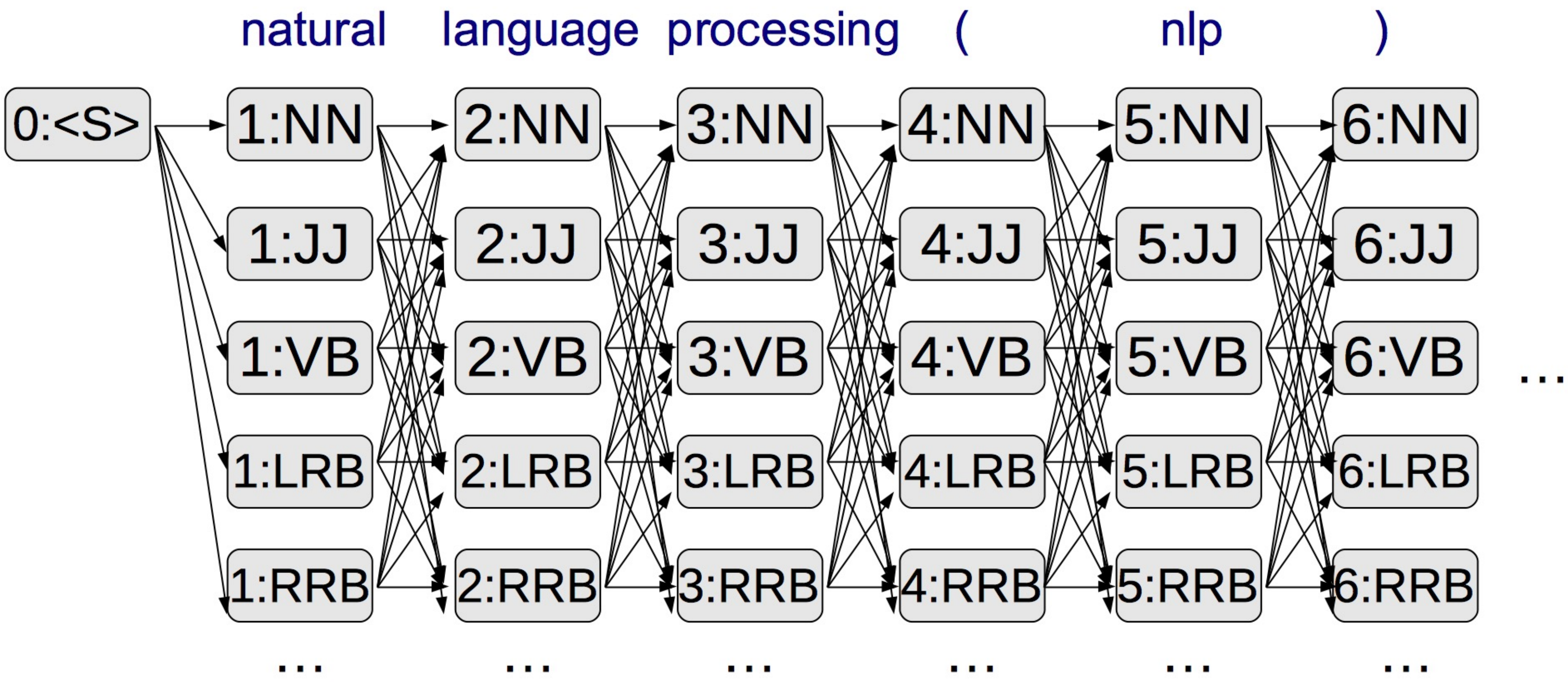
$$P(w_1 \dots w_n | t_1 \dots t_n) \approx \prod_{i=1}^{n} P(w_i | t_i)$$

Markov (bigram) assumption

$$P(t_1 \dots t_n) \approx \prod_{i=1}^{n} P(t_i | t_{i-1})$$

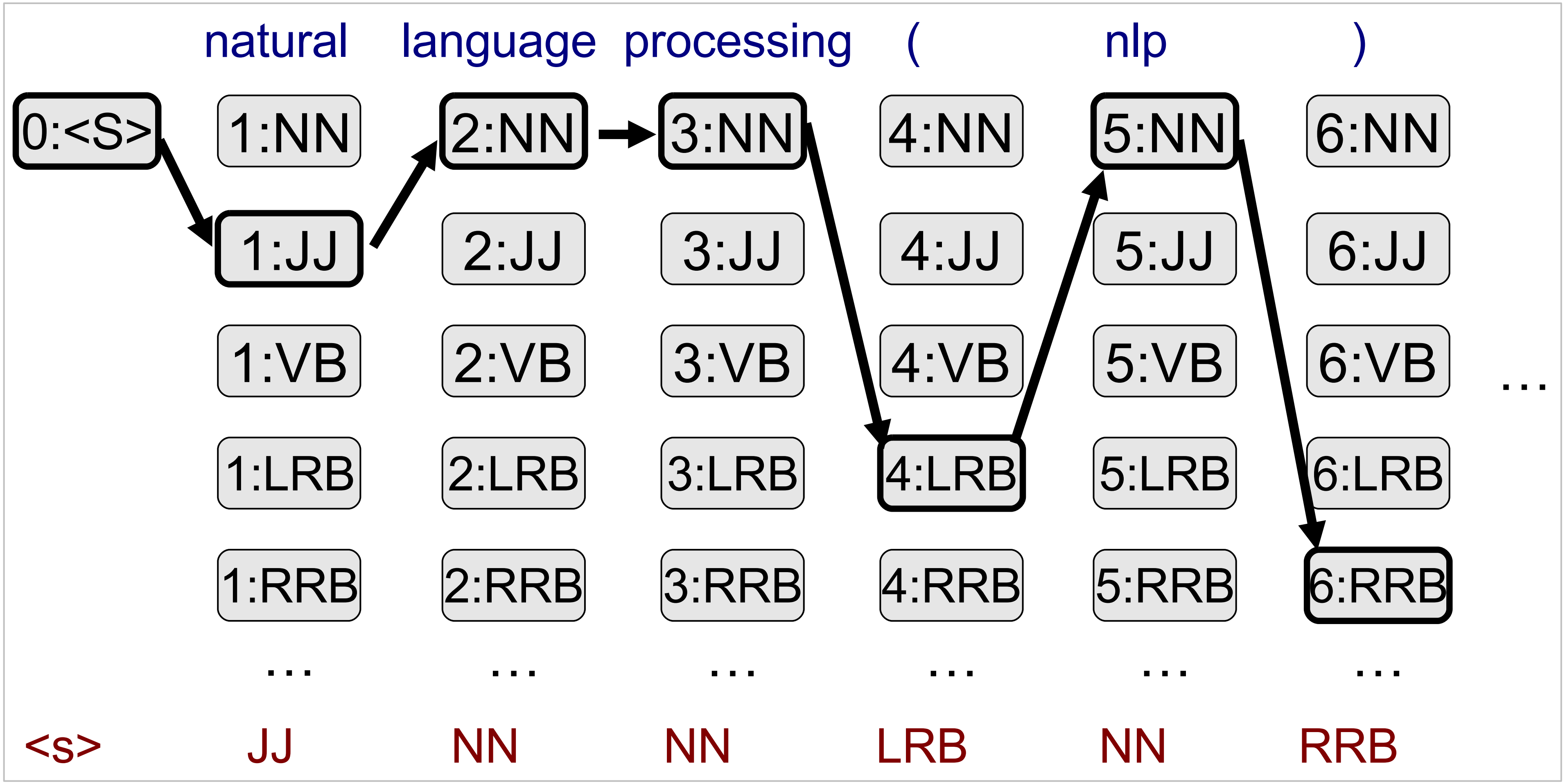
$$\hat{t}_{1:n} = \underset{t_1 \dots t_n}{\operatorname{argmax}} P(t_1 \dots t_n | w_1 \dots w_n) \approx \underset{t_1 \dots t_n}{\operatorname{argmax}} \prod_{i=1}^{n} \overbrace{P(w_i | t_i)}^{\text{emission}} \overbrace{P(t_i | t_{i-1})}^{\text{transition}}$$

Finding POS tags with Markov Models



Finding POS Tags with Markov Models

- The best path is our POS sequence



Viterbi algorithm

- At each cell, $v_t(j)$ represents the highest probability for any sequence $q_1 \dots q_t$ ending at the state j

$$v_t(j) = \max_{q_1,\dots,q_{t-1}} P(q_1 \dots q_{t-1}, o_1, o_2, \dots o_t, q_t = j|\lambda)$$

  - λ represents the HMM model

Viterbi algorithm

- Dynamic programming

$$v_t(j) = \max_i v_{t-1}(i) a_{ij} b_j(o_t)$$

$v_{t-1}(i)$	the **previous Viterbi path probability** from the previous time step
a_{ij}	the **transition probability** from previous state q_i to current state q_j
$b_j(o_t)$	the **state observation likelihood** of the observation symbol o_t given the current state j

Viterbi algorithm

- Dynamic programming

$$v_t(j) = \max_i v_{t-1}(i) a_{ij} b_j(o_t)$$

- In implementation, we will use negative logarithm to avoid underflow problem
- The score of the best path upto the step t and ends with the state j is denoted by $v'_t(j) = -\log v_t(j)$

$$\begin{aligned} &v'_t(j) \\ &= \min_i[-\log v_{t-1}(i) + \ -\log a_{ij} + \ -\log b_j\,(o_t)] \\ &= \min_i[v'_{t-1}(i) + \ -\log a_{ij} + \ -\log b_j\,(o_t)] \end{aligned}$$

Viterbi Algorithm Steps

- Forward step, calculate the best path to a node
  - Find the path to each node with the lowest negative log probability
- Backward step, reproduce the path

Forward Step: Part 1

- First, calculate transition from <S> and emission of the first word for every POS

natural

0:<S>

1:NN best_score[“1 NN”] = -log P_T(NN|<S>) + -log P_E(natural | NN)

1:JJ best_score[“1 JJ”] = -log P_T(JJ|<S>) + -log P_E(natural | JJ)

1:VB best_score[“1 VB”] = -log P_T(VB|<S>) + -log P_E(natural | VB)

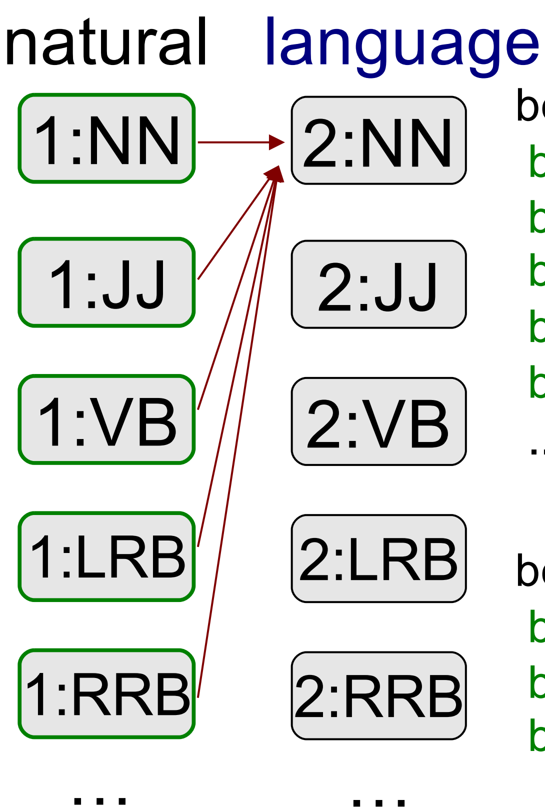
1:LRB best_score[“1 LRB”] = -log P_T(LRB|<S>) + -log P_E(natural | LRB)

1:RRB best_score[“1 RRB”] = -log P_T(RRB|<S>) + -log P_E(natural | RRB)

…

Forward Step: Middle Parts

- For middle words, calculate the minimum score for all possible previous POS tags

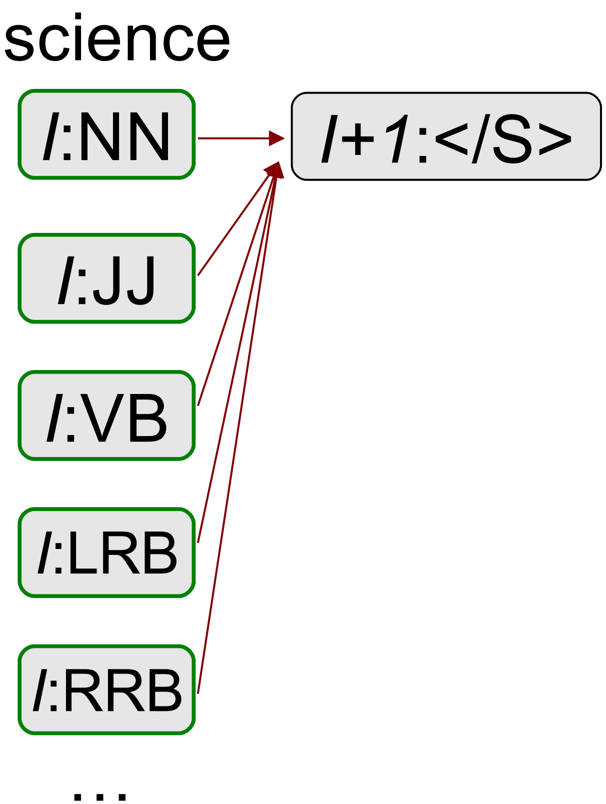


best_score["2 NN"] = min(
best_score["1 NN"] + -log P_T (NN|NN) + -log P_E(language | NN),
best_score["1 JJ"] + -log P_T (NN|JJ) + -log P_E (language | NN),
best_score["1 VB"] + -log P_T(NN|VB) + -log P_E(language | NN),
best_score["1 LRB"] + -log P_T(NN|LRB) + -log P_E (language | NN),
best_score["1 RRB"] + -log P_T (NN|RRB) + -log P_E(language | NN),
...)

best_score["2 JJ"] = min(
best_score["1 NN"] + -log P_T(JJ|NN) + -log P_E(language | JJ),
best_score["1 JJ"] + -log P_T(JJ|JJ) + -log P_E(language | JJ),
best_score["1 VB"] + -log P_T(JJ|VB) + -log P_E(language | JJ),
...

Forward Step: Final Part

- Finish up the sentence with the sentence final symbol



best_score["*l*+*1* </S>"] = min(
  best_score["*l* NN"] + -log P_T(</S>|NN),
  best_score["*l* JJ"] + -log P_T(</S>|JJ),
  best_score["*l* VB"] + -log P_T(</S>|VB),
  best_score["*l* LRB"] + -log P_T(</S>|LRB),
  best_score["*l* NN"] + -log P_T(</S>|RRB),
  ...
)

Implementation: Model Loading

make a map for *transition, emission, possible_tags*

for each *line* **in** model_file
    split *line* **into** *type, context, word, prob*
    possible_tags[context] = 1 # We use this to
                                    # enumerate all tags
    if *type* = “T”
        transition[“*context word*”] = *prob*
    else
        emission[“*context word*”] = *prob*

Implementation: Forward Step

split *line* **into** *words*
I = length(*words*)
make maps *best_score*, *best_edge*
best_score["0 <s>"] = 0   # Start with <s>
best_edge["0 <s>"] = NULL
for *i* **in** 0 … *I*-1:
   for each *prev* **in** keys of *possible_tags*
      for each *next* **in** keys of *possible_tags*
         if best_score["*i prev*"] **and** transition["prev next"] **exist**
            score = best_score["i prev"] +
                           -log P_T(next|prev) + -log P_E(word[i]|next)
            if *best_score*["*i+1 next*"] **is** new **or** > *score*
               best_score["*i+1 next*"] = score
               best_edge["*i+1 next*"] = "*i prev*"
Finally, do the same for </s>

Implementation: Backward Step

tags = [ ]
next_edge = *best_edge*[*"I </s>"*]
while *next_edge* != "0 <s>"
    # Add the substring for this edge to the words
    split *next_edge* **into** *position, tag*
    append *tag* **to** *tags*
    next_edge = *best_edge*[*next_edge*]
tags.reverse()
join *tags* into a string and **print**

Lecture outline

- Sequence Labeling Problems
- Hidden Markov Models for Part-of-Speech Tagging
- Conditional Random Fields
- Evaluation of Named Entity Recognition

Problems of HMM tagger

- Unknown words
  - Proper names and accronyms
  - New common verbs and nouns
- Difficult to incoporate attribiary features to HMM

Conditional Random Fields (CRF)

- A discriminative sequence model based on log-linear models

Given $X = x_1^n = x_1 \ldots x_n$. We want to compute sequence of output tags $Y = y_1^n = y_1 \ldots y_n$

- CRF computes posterior probability $P(Y|X)$ directly

$$\hat{Y} = \underset{Y}{\arg\max}\, P(Y|X)$$

Conditional Random Fields

- CRF models $P(Y|X)$ by using feature functions f

$$p(Y|X) = \frac{\exp\left(\sum_{k=1}^{K} w_k F_k(X,Y)\right)}{\sum_{Y' \in \mathcal{Y}} \exp\left(\sum_{k=1}^{K} w_k F_k(X,Y')\right)}$$

- We call $F_k(X,Y)$ **global features**
  - Each one is a property of the entire input sequence X and output sequence Y

Global Features

- Decompose global features into a sum of local features for each position i in Y

$$F_k(X,Y) = \sum_{i=1}^{n} f_k(y_{i-1}, y_i, X, i)$$

Features in a CRF POS Tagger

- Each local feature depends on any information from (y_{i-1}, y_i, X, i)
  - $I(x_i = the, y_i = \mathrm{DET})$
  - $I(y_i = \mathrm{PROPN}, x_{i+1} = Street, y_{i-1} = \mathrm{NUM})$
  - $I(y_i = \mathrm{VERB}, y_{i-1} = \mathrm{AUX})$

  - $I\{x\}$ is an indicator function
    - 1 if x is true, 0 otherwise

Features in a CRF POS Tagger

- Feature templates
  - $\langle y_i, x_i \rangle, \langle y_i, y_{i-1} \rangle, \langle y_i, x_{i-1}, x_{i+2} \rangle$
- Features generated for the example *Janet/NNP will/MD back/VB the/DT bill/NN* when x_i is the word *back*
  - f_{2341}: y_i = VB and x_i = back
  - f_{100}: y_i = VB and y_{i-1} = MD
  - f_{99451}: y_i = VB and x_{i-1} = will and x_{i+2} = bill

Features in CRF POS Tagger

- Features that help with unknown words
  - Word shape features
  - Prefix and suffix features
- E.g., features for the word Hà_Nội

  prefix(x_i) = H

  prefix(x_i) = Hà

  suffix(x_i) = ội

  suffix(x_i) = i

  word-shape(x_i) = Xx_Xxx

  short-word-shape(x_i) = Xx_Xx

Features for CRF Named Entity Recognzers

- Features are very similar features to a POS tagger

identity of w_i, identity of neighboring words
embeddings for w_i, embeddings for neighboring words
part of speech of w_i, part of speech of neighboring words
presence of w_i in a gazetteer
w_i contains a particular prefix (from all prefixes of length 4)
w_i contains a particular suffix (from all suffixes of length 4)
word shape of w_i, word shape of neighboring words
short word shape of w_i, short word shape of neighboring words
gazetteer features

Features for CRF Named Entity Recognzers

- E.g., features for entity token *L'Occitane*

prefix(x_i) = L suffix(x_i) = tane
prefix(x_i) = L' suffix(x_i) = ane
prefix(x_i) = L'O suffix(x_i) = ne
prefix(x_i) = L'Oc suffix(x_i) = e
word-shape(x_i) = X'Xxxxxxxx short-word-shape(x_i) = X'Xx

Inference and Training for CRFs

$$
\begin{aligned}
\hat{Y} &= \underset{Y \in \mathcal{Y}}{\operatorname{argmax}} P(Y|X) \\
&= \underset{Y \in \mathcal{Y}}{\operatorname{argmax}} \frac{1}{Z(X)} \exp\left(\sum_{k=1}^{K} w_k F_k(X,Y)\right) \\
&= \underset{Y \in \mathcal{Y}}{\operatorname{argmax}} \exp\left(\sum_{k=1}^{K} w_k \sum_{i=1}^{n} f_k(y_{i-1}, y_i, X, i)\right) \\
&= \underset{Y \in \mathcal{Y}}{\operatorname{argmax}} \sum_{k=1}^{K} w_k \sum_{i=1}^{n} f_k(y_{i-1}, y_i, X, i) \\
&= \underset{Y \in \mathcal{Y}}{\operatorname{argmax}} \sum_{i=1}^{n} \sum_{k=1}^{K} w_k f_k(y_{i-1}, y_i, X, i)
\end{aligned}
$$

Viterbi equation for CRFs

Use Viterbi algorithm in decoding

$$v_t(j) = \max_{i=1\ldots N} v_{t-1}(i) \sum_{k=1}^{K} w_k f_k(y_{t-1}, y_t, X, t)$$

$$1 \leq j \leq N, 1 < t \leq T$$

Learning in CRFs relies on the same supervised learning algorithm presented in logistic regression

- Stochastic Gradient Descent

Libraries that implement CRFs

- C++
  - CRF++
  - CRFSuite
- Mallet
- Python Wrappers
  - sklearn-crfsuite
  - python-crfsuite
  - Python wrapper in CRF++

Lecture outline

- Sequence Labeling Problems
- Hidden Markov Models for Part-of-Speech Tagging
- Conditional Random Fields
- Evaluation of Named Entity Recognition

Evaluation

- Part-of-Speech tagging is evaluated by accuracy
  - Ratio of correct labeled tags to total tags

- Named Entity Recognition is evaluated by Precision, Recall, F1
  - Recall is the ratio of the number of correctly labeled responses to the total that should have been labeled;
  - Precision is the ratio of the number of correctly labeled responses to the total labeled;
  - F -measure is the harmonic mean of the two.

- We often use seqeval for evaluating sequence labeling tasks