NER Project (Programming Practice)

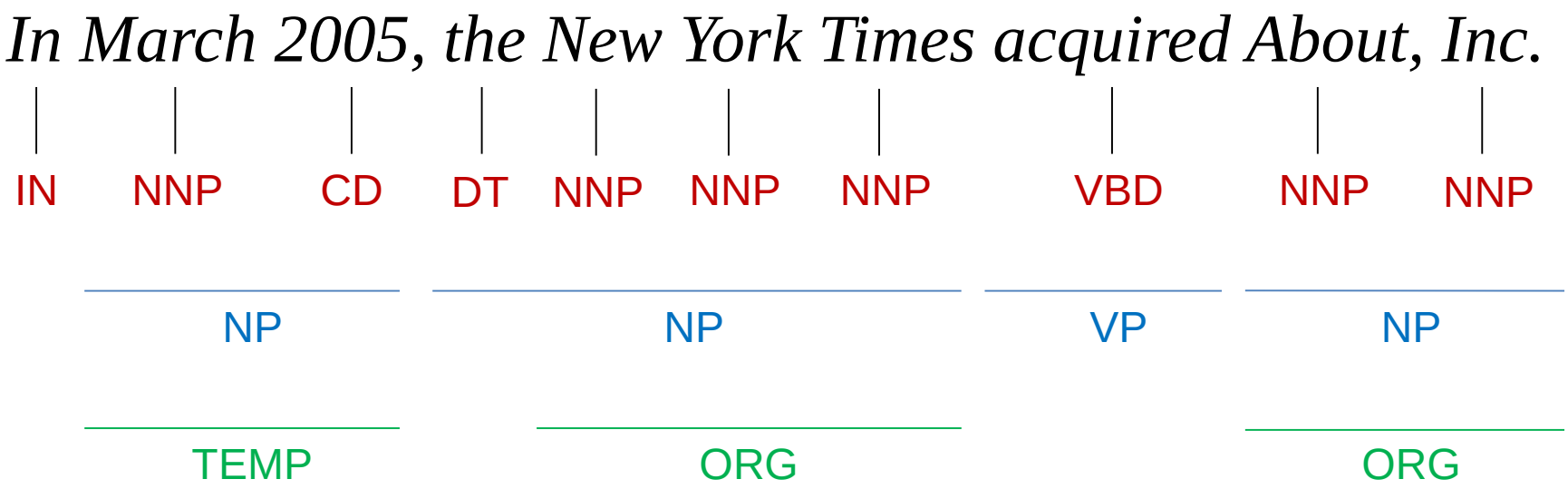
Naoaki Okazaki (okazaki at ecei.tohoku.ac.jp),
Kentaro Inui (inui at ecei.tohoku.ac.jp)

Goal of this project

- Build a high-performance named entity recognizer (NER)
  - Who builds the best performing NER?
  - How do they achieve?
- Exercise with a real data set
  - *Experiencing (seeing) is believing!*
- Learn the difficulty in solving an NLP task
  - Analyze why your NER cannot be perfect
  - Where does the difficulty actually come?

Named Entity Recognition (NER)

- Identify phrases that refer to entities:
  - For example, people, organizations, locations, time



- In this project, part-of-speech and chunk tags are given
- Predict named entity spans and types

Why do we care NEs?

- Subtask of extracting knowledge from text

In March 2005, the New York Times acquired About, Inc.

TEMP ORG ORG

Template extraction

<ORG> *acquire* <ORG>

Information extraction

Event	Agent	Theme	Time
acquisition	New York Times	About, Inc.	March 2005
…	…	…	…
…	…	…	…

CoNLL 2003 data set

- Concentrate on four types of NEs
  - PER (people), LOC (location), ORG (organization),
  - MISC (miscellaneous): e.g., country names
- Domain
  - News (Reuters corpus)
- Size
  - Train: 14,041 sentences; 34,043 / 203,621 (16.7%) NE tokens
  - Dev: 3,250 sentences; 8,603 / 51,362 (16.7%) NE tokens
  - Test: 3,453 sentences; 8,112 / 46,435 (17.5%) NE tokens

Example of the data set

- Each line consists of NE, token, POS, and chunk strings separated by TAB characters
- Segments (spans) of NEs and chunks are represented by IOB2 notation
- An empty line represents an end of sentence

NE	Token	POS	Chunk
B-LOC	Italy	NNP	B-NP
O	recalled	VBD	B-VP
B-PER	Marcello	NNP	B-NP
I-PER	Cuttitta	NNP	I-NP

← Empty line

Various approaches to NER

- Dictionary-based
  - Look-up a dictionary (gazetteer) containing NE expressions
    - Mark “Italy” as B-LOC if it is included in a ‘location’ dictionary
- Rule-based (*before machine learning*)
  - The pattern “*ddd-dddd*” presents a postal code (*d*: digit)
  - A token starting with a capital letter is expected to be an NE
  - Difficult to design the best combination of rules
- Machine-learning based (*most popular*)
  - Similar to rule-based approach, but rule weights are automatically tuned by a training set
  - NER can be formalized as a sequential labeling problem

NER as Sequential Labeling Problem

(Recap of sequential labeling problem)

Sequential Labeling Problem

- A given sentence, *"Italy recalled Marcello Cuttitta"*
- Represent the input sentence with a vector $\boldsymbol{x}$
- Predict NE tags (a vector $\boldsymbol{y}$) for the input $\boldsymbol{x}$

t	1	2	3	4	$(\mathrm{T}=4)$
$\boldsymbol{x}$	$Italy$	$recalled$	$Marcello$	$Cuttitta$	
	x_1	x_2	x_3	x_4	
$\boldsymbol{y}$	B_LOC	O	B_PER	I_PER	
	y_1	y_2	y_3	y_4	

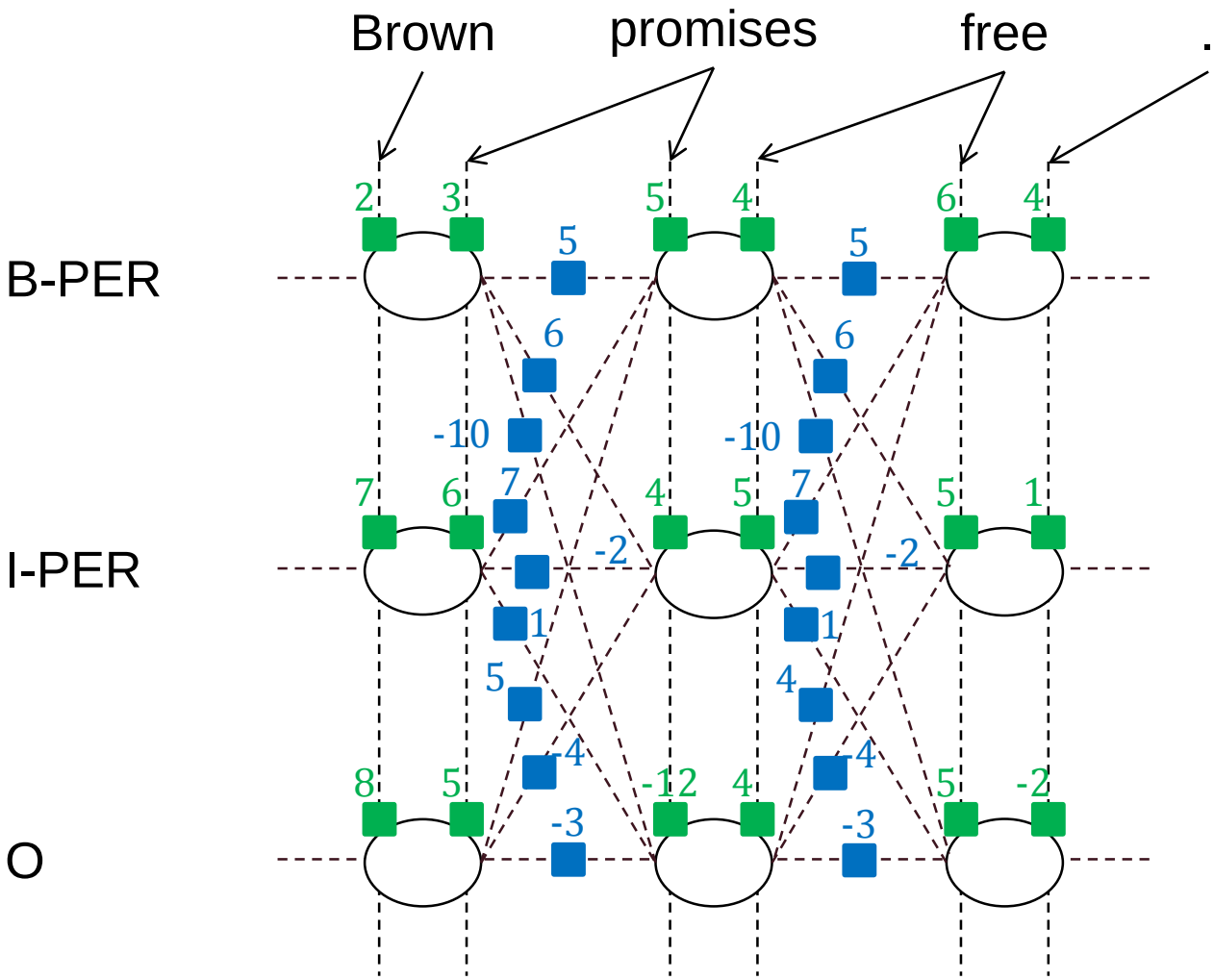
Predict

$$\hat{\boldsymbol{y}} = \underset{\boldsymbol{y}}{\operatorname{argmax}} P(\boldsymbol{y}|\boldsymbol{x})$$

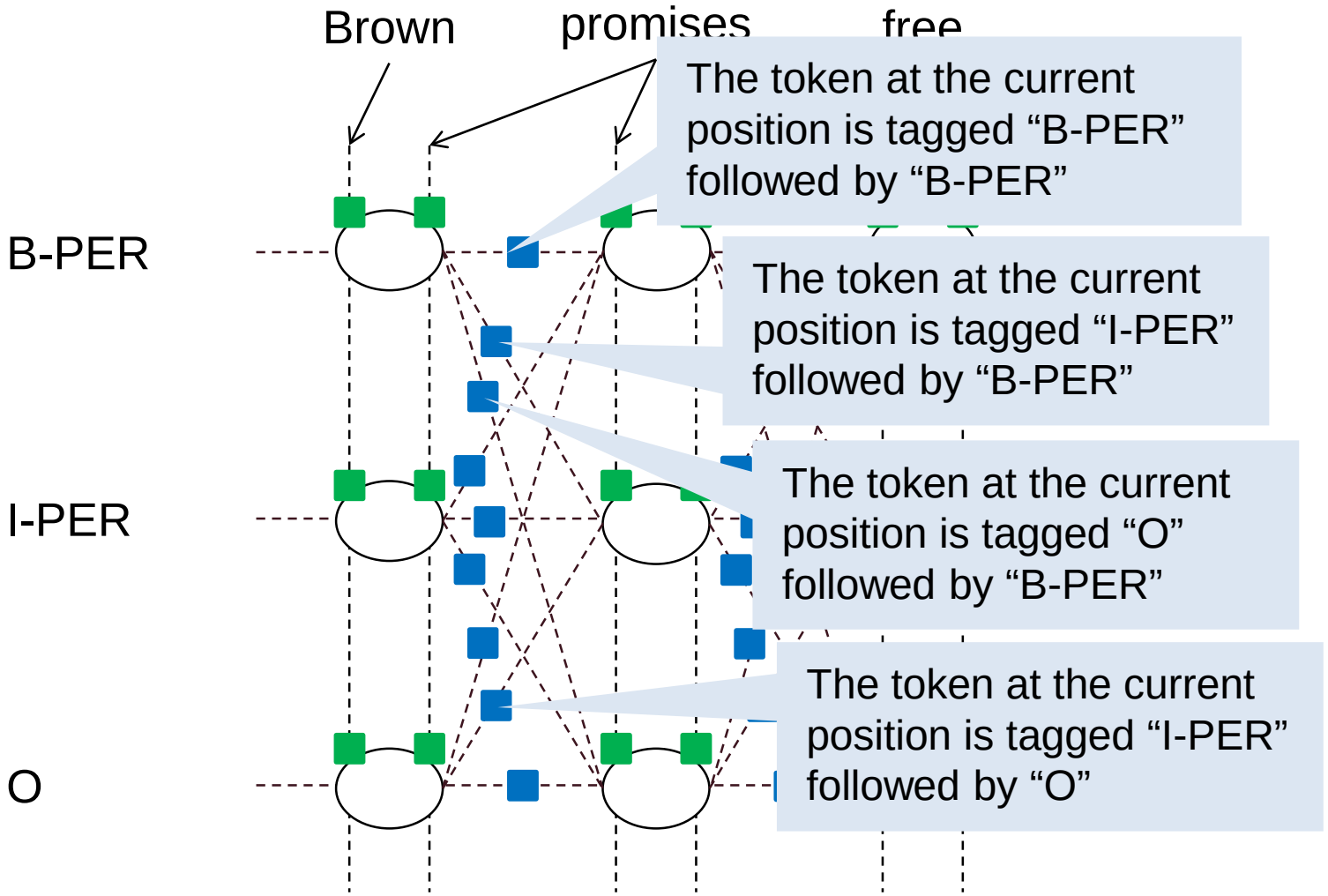
$\boldsymbol{y_1}, \boldsymbol{y_2}, \boldsymbol{y_3}, \cdots$

$\hat{\boldsymbol{y}}$ means "our estimation for $\boldsymbol{y}$"
argmax: find $\boldsymbol{y}$ that maximizes $P(\boldsymbol{y}|\boldsymbol{x})$

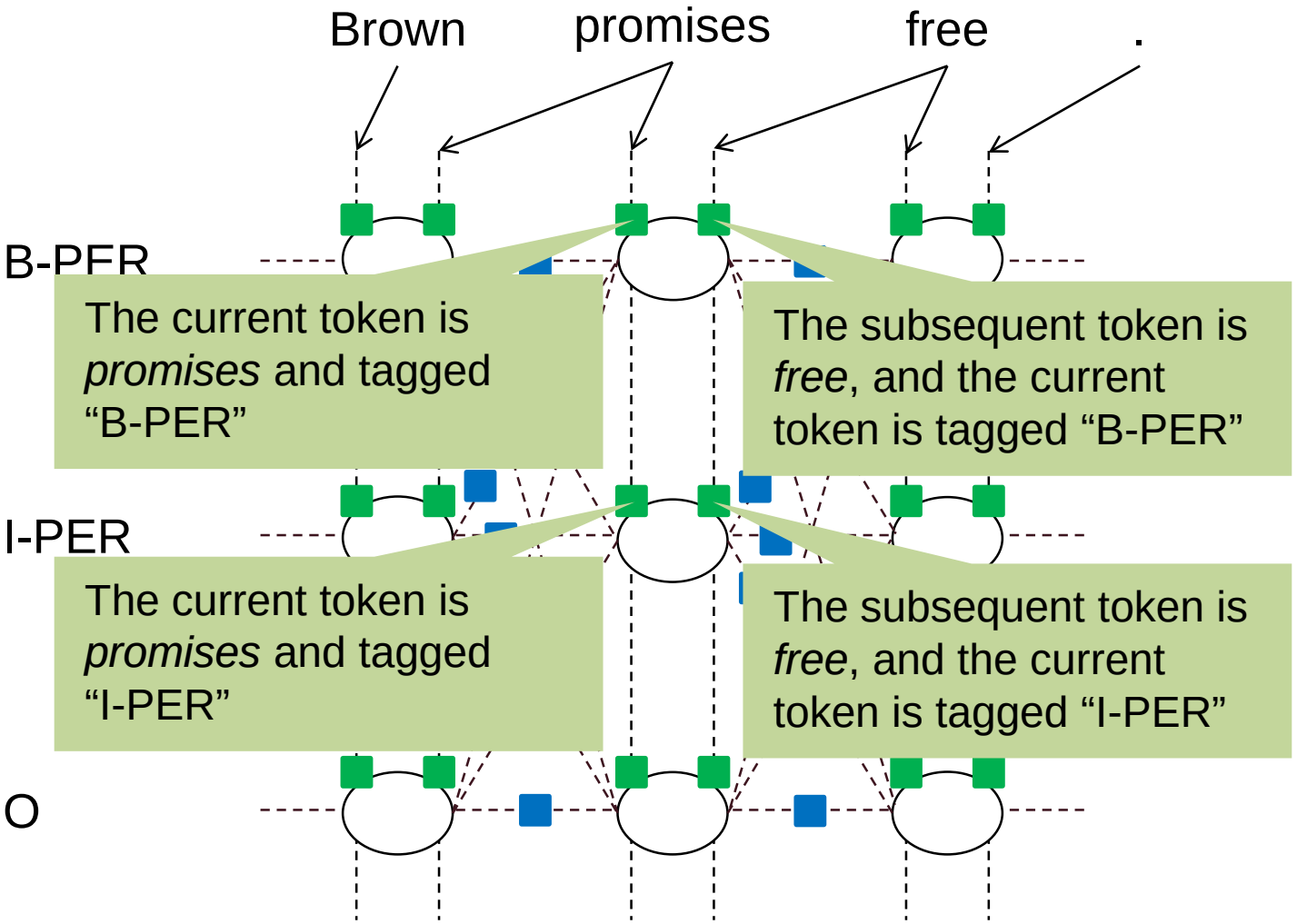
Lattice representation of structured perceptron model (This is a simplified example for explanation)



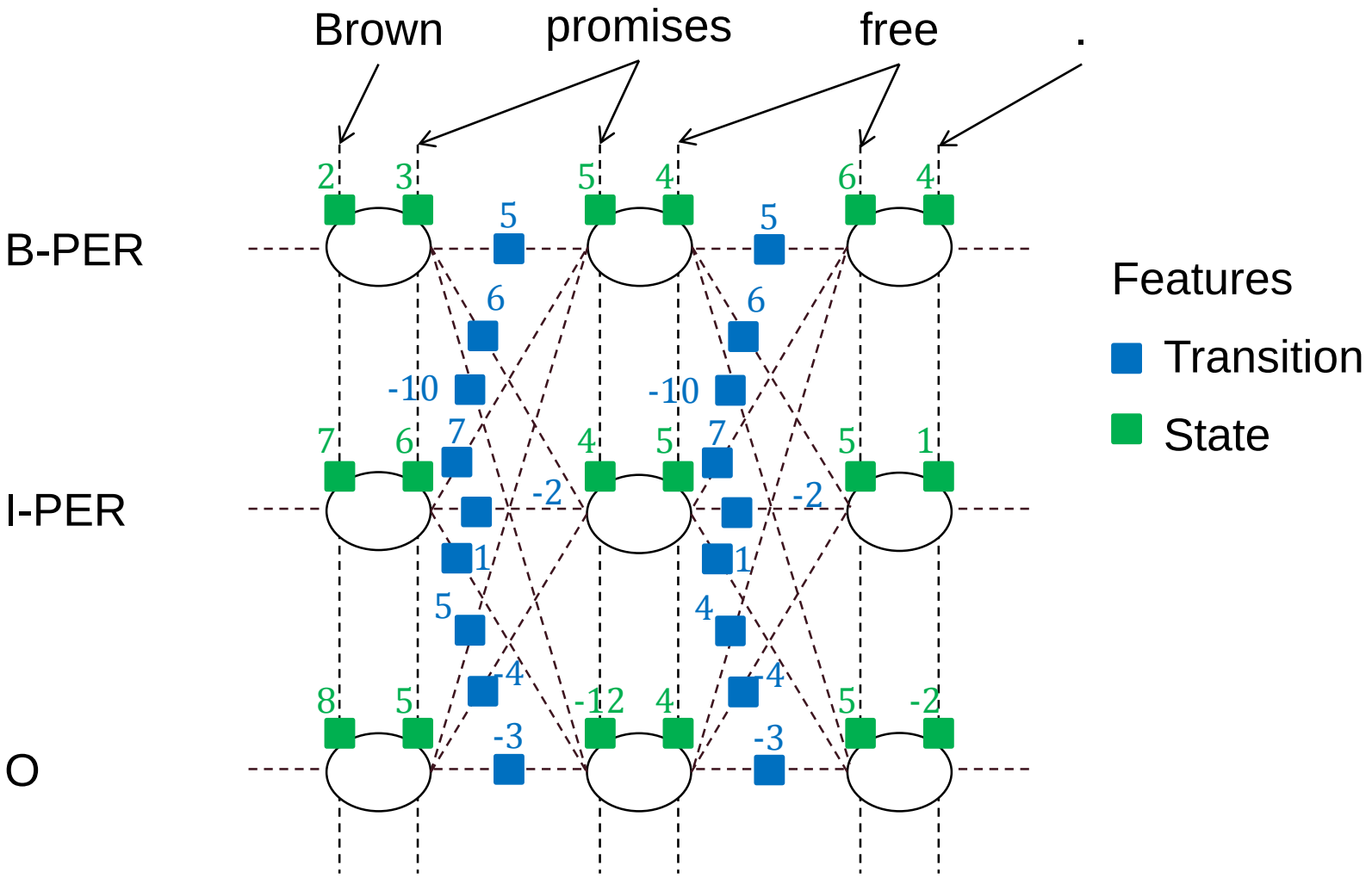
Transition (label-bigram) features



State (unigram) features

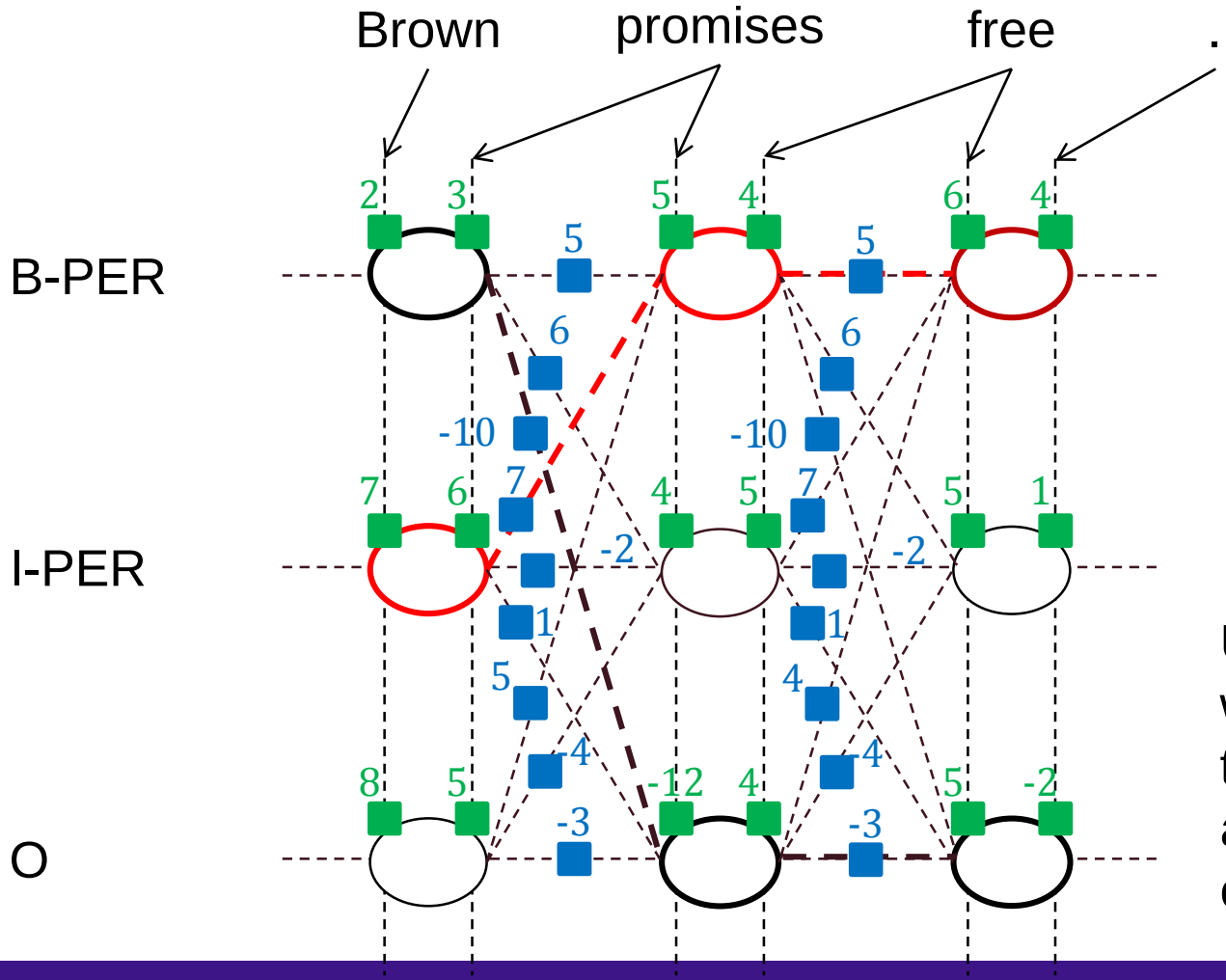


Lattice representation



Training

- $(\boldsymbol{x}_n, \boldsymbol{y}_n) = (\text{Brown promises change, B}-\text{PER O O})$



Viterbi path

Correct path

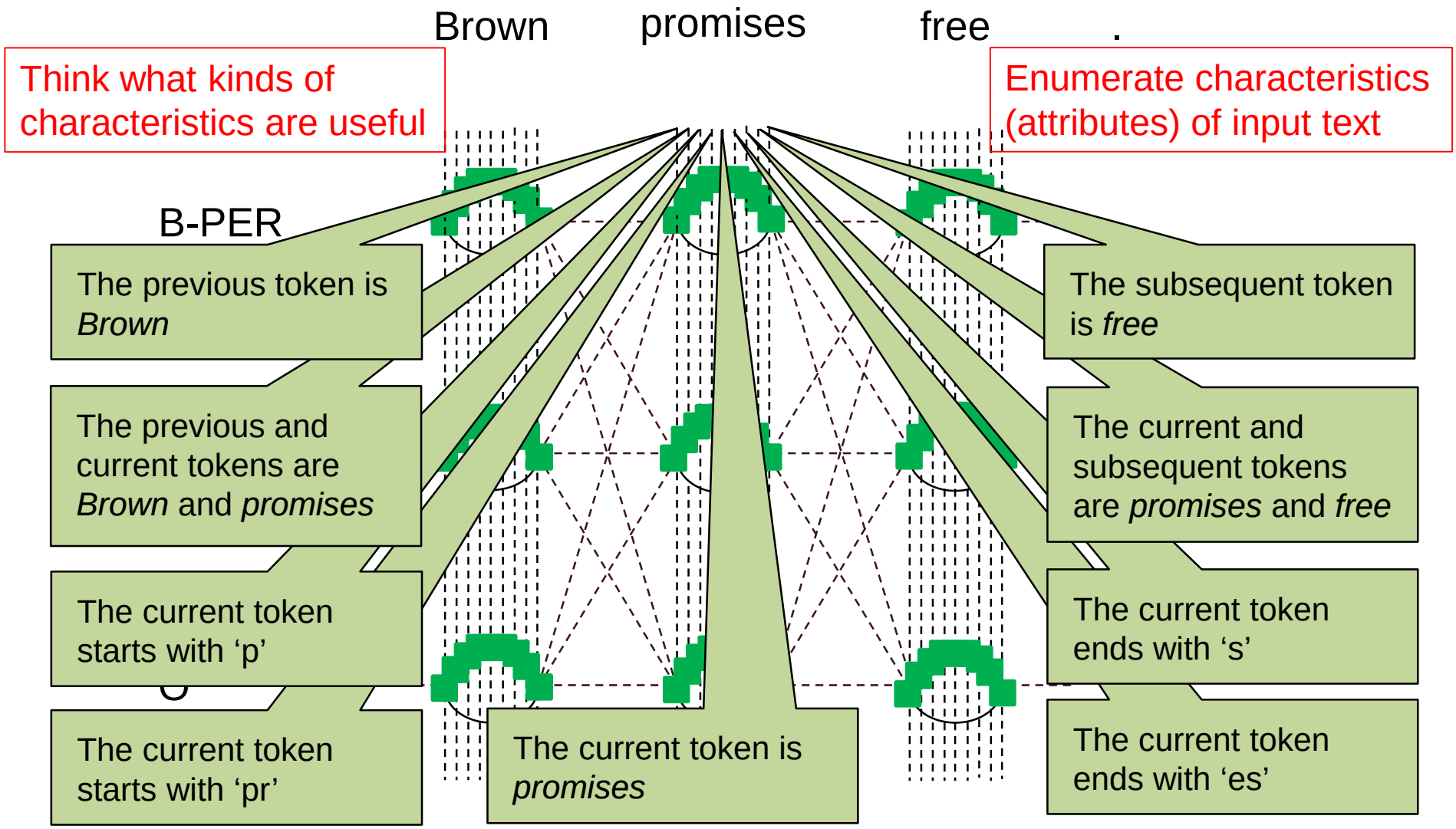
Update feature weights such that the Viterbi path agrees with the correct path

What is necessary for building an NER?

- Implemented by an existing tool
  - Training algorithm (data I/O, Viterbi, weight update)
  - Transition features (generated automatically by the tool)

- Not implemented by an existing tool
  - State features
    - Extracting characteristics of the input that may be useful for NER
    - We need to *design* and implement state features (feature engineering)

More concretely…



Tutorial on building an NER

Attribute (feature) extraction

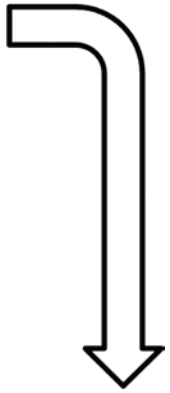
- State feature in this tutorial
  - Token unigrams and bigrams in [-2, +2] window
  - POS unigrams and bigrams in [-2, +2] window
  - Chunk unigrams and bigrams in [-2, +2] window
  - Unigrams and bigrams of initial-upper patterns in [-2, +2] window

- Attribute extractor (feature.py)
  - Convert training and development sets into attribute data

```
$ ./feature.py < train > train.f
$ ./feature.py < dev > dev.f
```

Attributes for the token “German”

```
B-ORG    EU       NNP   B-NP
O        rejects  VBZ   B-VP
B-MISC   German   JJ    B-NP
O        call     NN    I-NP
O        to       TO    B-VP
O        boycott  VB    I-VP
B-MISC   British  JJ    B-NP
O        lamb     NN    I-NP
O        .        .     O
```



- Token unigrams and bigrams
- POS unigrams and bigrams
- Chunk unigrams and bigrams
- Initial-upper patterns

```
w[-2]=EU w[-1]=rejects w[0]=German w[1]=call w[2]=to w[-2]|w[-
1]=EU|rejects w[-1]|w[0]=rejects|German w[0]|w[1]=German|call
w[1]|w[2]=call|to pos[-2]=NNP pos[-1]=VBZ pos[0]=JJ pos[1]=NN
pos[2]=TO pos[-2]|pos[-1]=NNP|VBZ pos[-1]|pos[0]=VBZ|JJ
pos[0]|pos[1]=JJ|NN pos[1]|pos[2]=NN|TO chk[-2]=B-NP chk[-1]=B-VP
chk[0]=B-NP chk[1]=I-NP chk[2]=B-VP chk[-2]|chk[-1]=B-NP|B-VP
chk[-1]|chk[0]=B-VP|B-NP chk[0]|chk[1]=B-NP|I-NP chk[1]|chk[2]=
iu[-2]=True     iu[-1]=False    iu[0]=True      iu[1]=False
iu[2]=False     iu[-2]|iu[-1]=True|False        iu[-
1]|iu[0]=False|True iu[0]|iu[1]=True|False
iu[1]|iu[2]=False|False
```

Training

- Train a model using structured perceptron
  - The fastest algorithm for training a model
- You don’t have to implement the perceptron algorithm
  - Simply use an existing tool (as most NLP researchers do)
- Use CRFsuite (http://www.chokkan.org/software/crfsuite/)
  - Training data: train.f
  - Model (feature weights): ner.model
  - Training algorithm: averaged perceptron (20 iterations)

```
$ crfsuite learn -a ap -p max_iterations=20 -m ner.model train.f
```

Tagging

- Apply the model (ner.model) to predict NE labels of the test (development) data
  - Run Viterbi algorithm with the feature weights

```
$ crfsuite tag -m ner.model < dev.f > dev.tagged
```

- For evaluation, it is convenient to keep the correct labels in the test data
  - Output the correct and predicted labels in the test data

```
$ crfsuite tag –r -m ner.model < dev.f > dev.eval
```

Evaluation

- Use the development set (dev) for improving your NER
  - Do not use the training set (train) for evaluation

- Measure the performance using conlleval.py

```
$ conlleval.py < dev.eval
```

Analysis

- Write a program that output sentences with false (incorrect) predictions

```
B-ORG   B-ORG   EU       NNP  B-NP
O       O       rejects  VBZ  B-VP
B-ORG   B-MISC  German   JJ   B-NP
O       O       call     NN   I-NP
O       O       to       TO   B-VP
O       O       boycott  VB   I-VP
B-ORG   B-MISC  British  JJ   B-NP
O       O       lamb     NN   I-NP
O       O       .        .    O
```

  - Useful for identifying possible reasons why a tagger was confused

How to add an attribute

- Add an attribute of tokens
  - `v['iu'] = str(v['w'] and v['w'][0].isupper())`

- Register the attribute to the feature template
  - Unigram: `templates += [(('iu', i),) for i in range(-2, 3)]`
  - Bigram: `templates += [(('iu', i), ('iu', i+1)) for i in range(-2, 2)]`

Hints for high-performance NER

Attributes (1/3)

Name	Description	Feature value
w	Token	Peter
wl	Lowercased token	peter
lem	Lemma of the lowercased token	peter
pos	POS of the token	NNP
chk	Chunk tag of the token	B-NP
shape	Character shape	ULLLL
shaped	Designated character shape	UL
type	Character type	InitUpper
p*n* (*n*=1…4)	Prefix (of length *n*) of the token	(P, Pe, Pet, Pete)
s*n* (*n*=1…4)	Suffix (of length *n*) of the token	(r, er, ter, eter)

Character type: {AllUpper, AllDigit, AllSymbol, AllUpperDigit, AllUpperSymbol, AllDigitSymbol, AllUpperDigitSymbol, InitUpper, AllLetter, AllAlnum}

Attributes (2/3)

Name	Description	True token
2d	Two digits	11
4d	Four digits	2011
d&a	Digits and alphabets	TTF1
d&-	Digits and ‘-’	03-5841-4120
d&/	Digits and ‘/’	2011/01/20
d&,	Digits and ‘,’	1,2
d&.	Digits and ‘.’	1.2
up	Capital and period	A.
date	Date expression	2011-12-12

Each feature in this table yields a Boolean value (‘yes’ or ‘no’).

Attributes (2/3)

Name	Description	True token
iu	Initial upper	Peter
au	All upper	EU
al	All lower	love
ad	All digits	2011
ao	All others (non-alphanumerics)	$
cu	Contain upper	Peter
cl	Contain lower	Peter
ca	Contain alphabets	Peter
cd	Contain digits	TTF1
cs	Contain others (non-alphanumerics)	ADRB1/ADRB2

Each feature in this table yields a Boolean value (‘yes’ or ‘no’).

Gazetteers (dictionaries)

- Gazetteers for CoNLL 2003 (KnownLists)
  - Obtained from the one bundled in LBJ NER tagger
    - 14 high-precision low-recall lists extracted from the Web
    - 16 lists extracted from WikiPedia

- Matching methods:
  - Case-sensitive exact matching
  - Case-insensitive exact matching
  - Case-insensitive partial matching
    - Fire if the token is included any of gazetteer entries

Cluster features

- Clustering results of nouns (brownBllipClusters)
- Bit-pattern representation of clusters
  - Japan: 011001011010
  - China: 011001011010
  - UK: 011011101
- We use prefixes of bit-patterns of length 4, 6, 10, 20
  - Japan: 0110, 011001, 0110010110, 011001011010
  - China: 0110, 011001, 0110010110, 011001011010
  - UK: 0110, 011011, 011011101

Belong to the same cluster

Belong to the same cluster

Tune the number of iterations

- Find the number of iterations where the obtained model performs the best on the development set
  - This is the only parameter to be tuned for averaged perceptron
- In order to do this…
  - Perform evaluations on models trained on different values of max_iterations (from 1 to 100)
  - This is time consuming!
- CRFsuite can measure the performance of a model at each iteration

```
$ crfsuite learn -a ap -p max_iterations=100 –e2 train.f dev.f
```

References

- Daniel M. Bikel, Richard Schwartz, Ralph M. Weischedel. An Algorithm that Learns What's in a Name. Machine Learning, 34(1-3), pp. 211-231, 1999.
- Lev Ratinov and Dan Roth. Design Challenges and Misconceptions in Named Entity Recognition. In Proc. of CoNLL '09, pages 147-155, 2009.

Course plan

- 2016-11-24 (today): explanation, preparation, …
- 2016-12-01: programing and analysis
- 2016-12-08: programing and analysis, intermediate championship
- 2016-12-15: No course given
- 2016-12-22: analysis, improving, writing a report