CS286 AI for Science and Engineering

Lecture 9: Natural Language Processing and Knowledge Representation

Kewei Tu (屠可伟)

PhD, Associate Professor

School of Information Science and Technology (SIST), ShanghaiTech University

Fall Semester, 2020

Outline

- Natural language processing (NLP)
  - Sequence labeling

- Knowledge representation (KR)
  - Propositional logic
  - First-order predicate logic

NLP – Sequence labeling

Sequence Labeling

- Problem Definition
  - Known
    - A set of labels $C = \{c_1, c_2, \ldots, c_J\}$
  - Input
    - Sentence s = $\{x^1, x^2, \ldots, x^m\}$
  - Output
    - For each word x^i, predict a label $c^i \in C$

Examples

- Part-of-speech tagging
  - Input

Pierre	Vinken	,	61	years	old	,	will	join	...

  - Output

NNP	NNP	,	CD	NNS	JJ	,	MD	VB

NNP = Proper noun, singular

CD = Cardinal number

NNS = Noun, plural

JJ = Adjective

...

Examples

- Chinese word segmentation
  - Input

    瓦 里 西 斯 的 船 只 中 …

  - Output

    B I I E S B E S

    (瓦 里 西 斯) (的) (船 只) (中) …

B = beginning of a word

I = inside of a word

E = end of a word

S = single character word

Examples

- Named entity recognition
  - Input

Michael	Jeffrey	Jordan	was	born	in	Brooklyn	...

  - Output

B-PER	I-PER	E-PER	O	O	O	S-LOC

Michael Jeffrey Jordan
Person

Brooklyn
Location

B = beginning of an entity
I = inside of an entity
E = end of an entity
S = single word entity
O = outside of any entity

-PER = person
-LOC = location
-ORG = organization
...

Examples

- Semantic role labeling
  - Input

The	cat	loves	hats	...

  - Output

B-ARG0	E-ARG0	S-PRED	S-ARG1

$\underline{\text{The cat}} \xleftarrow[\text{arg0}]{} \text{loves} \xrightarrow[\text{arg1}]{} \underline{\text{hats}}$

B = beginning of an entity

I = inside of an entity

E = end of an entity

S = single word entity

O = outside of any entity

-PRED = predicate

-ARG0 = agent

-ARG1 = patient

...

The simplest method

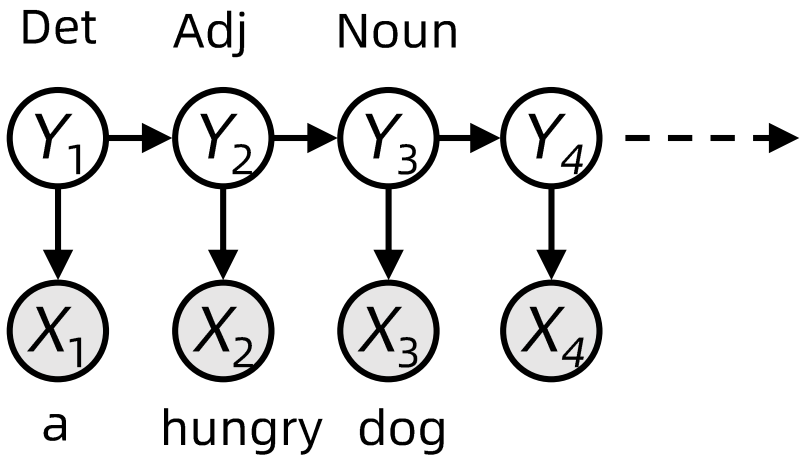
- For each word, predict its most frequent label
  - 90% accuracy on POS tagging!
  - Disadvantages:
    1. It does not consider the contextual info
       - “book a flight” vs. “read a book”
       - 我骑车差点摔倒，好在我一把把把把住了
    2. It does not consider relations between adjacent labels
       - In BIOES: "B-I" and "B-E" are OK, but "B-O" and "B-S" are not

Methods

- Hidden Markov Models
- Conditional Random Fields
- Recurrent Neural Networks

Hidden Markov Model (HMM)

- Variables
  - X: word
  - Y: label (hidden state)
- Parameters
  - Transition model $P(y_t|y_{t-1})$
  - Emission model $P(x_t|y_t)$
  - Initial distribution $P(y_1)$
    - Can be seen as transition from Y_0=START to Y_1
  - Final distribution $P(y_n)$
    - Can be seen as transition from Y_n to Y_{n+1}=STOP



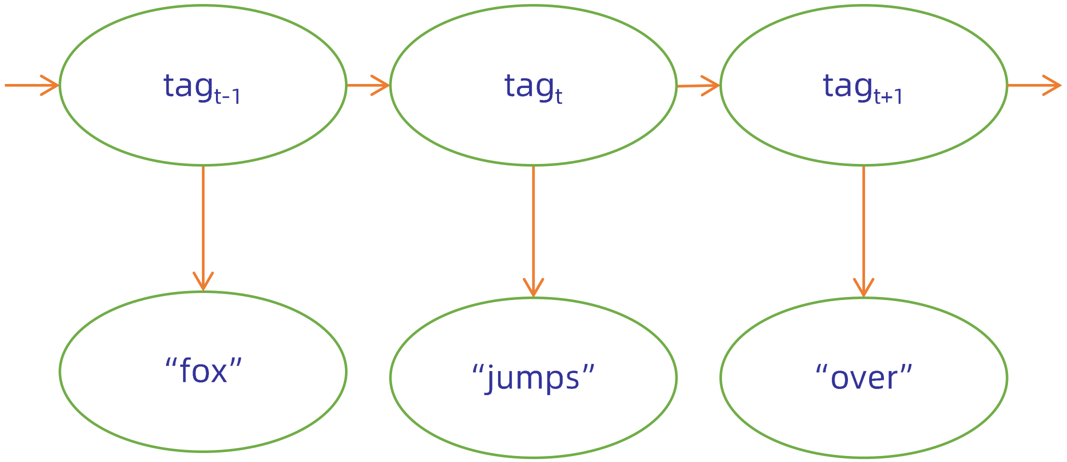
HMM Example

Transition

X_{t-1}	$P(X_t \mid X_{t-1})$			
	N	V	P	…
START	0.5	0.1	0.1	…
N	0.4	0.3	0.1	…
V	0.5	0	0.3	…
P	0.3	0.1	0	…
…	…	…	…	…

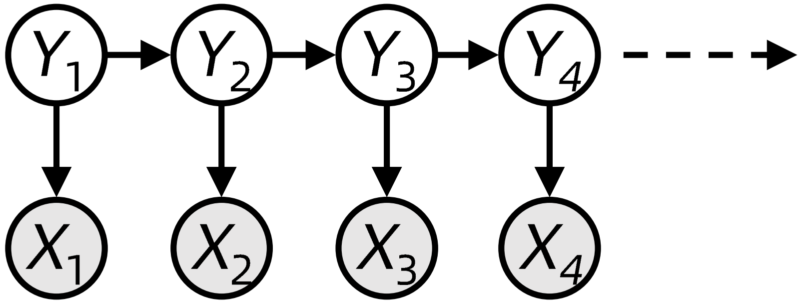
Emission

X_t	$P(E_t \mid X_t)$			
	“fox”	“dog”	“run”	…
N	0.02	0.03	0.01	…
V	0	0	0.05	…
P	0	0	0	…
…	…	…	…	…



HMM

- Joint distribution for hidden Markov model:
  - $P(Y_1, X_1, \ldots, YT, X_T) = \prod_{t=1:T} P(Y_t \mid Y_{t-1})\, P(X_t \mid Y_t)$
- Independence in HMM
  - Future states are independent of the past given the present
  - Current evidence is independent of everything else given the current state



HMM Inference

- Find the most likely label sequence of a sentence
- Inference objective

$$\hat{\mathbf{y}} = \arg\max_{y} P(\mathbf{y}|\mathbf{x}, A, B) = \arg\max_{\mathbf{y}} a_{START,y^1} \left(\prod_{t=1}^{|x|-1} b_{y^t,x^t} a_{y^t,y^{t+1}} \right) b_{y^{|x|},x^{|x|}} a_{y^{|x|},STOP}$$

- Inference Algorithm
  - Brute-force?
  - Viterbi

Viterbi Algorithm

a_{ij}	STOP	N	M	V	D
START	0	0.5	0	0.1	0.4
N	0.2	0.2	0.3	0.3	0
M	0	0	0	1	0
V	0.1	0.3	0.3	0	0.3
D	0	1	0	0	0

b_{ik}	tin	John	can	carried	a	...
N	0.05	0.01	0.01	0	0	...
M	0	0	0.2	0	0	...
V	0	0	0	0.02	0	...
D	0	0	0	0	0.5	...

i \ t	John_1	carried_2	a_3	tin_4	can_5
N					
M					
V					
D					

final

The probability of the most probable sequence up to t ending with a tag i

Viterbi Algorithm

a_{ij}	STOP	N	M	V	D
START	0	0.5	0	0.1	0.4
N	0.2	0.2	0.3	0.3	0
M	0	0	0	1	0
V	0.1	0.3	0.3	0	0.3
D	0	1	0	0	0

b_{ik}	tin	John	can	carried	a	...
N	0.05	0.01	0.01	0	0	...
M	0	0	0.2	0	0	...
V	0	0	0	0.02	0	...
D	0	0	0	0	0.5	...

Init: $$v_i^1 = a_{START,i} b_{i,x^1}, \quad i = 1, \ldots, N;$$

$i \backslash t$	John_1	$\mathrm{carried}_2$	a_3	tin_4	can_5
N					
M					
V					
D					

final

The probability of the most probable sequence up to *t* ending with a tag *i*

Viterbi Algorithm

a_{ij}	STOP	N	M	V	D
START	0	$\boxed{0.5}$	0	0.1	0.4
N	0.2	0.2	0.3	0.3	0
M	0	0	0	1	0
V	0.1	0.3	0.3	0	0.3
D	0	1	0	0	0

b_{ik}	tin	John	can	carried	a	...
N	0.05	$\boxed{0.01}$	0.01	0	0	...
M	0	0	0.2	0	0	...
V	0	0	0	0.02	0	...
D	0	0	0	0	0.5	...

Init: $v_i^1 = a_{START,i} b_{i,x^1}, \quad i = 1, \ldots, N;$

$i \backslash t$	John$_1$	carried$_2$	a$_3$	tin$_4$	can$_5$
N	0.005				
M					
V					
D					

final

The probability of the most probable sequence up to t ending with a tag i

Viterbi Algorithm

a_{ij}	STOP	N	M	V	D
START	0	0.5	0	0.1	0.4
N	0.2	0.2	0.3	0.3	0
M	0	0	0	1	0
V	0.1	0.3	0.3	0	0.3
D	0	1	0	0	0

b_{ik}	tin	John	can	carried	a	...
N	0.05	0.01	0.01	0	0	...
M	0	0	0.2	0	0	...
V	0	0	0	0.02	0	...
D	0	0	0	0	0.5	...

Init: $$v_i^1 = a_{START,i} b_{i,x^1}, \quad i = 1, \dots, N;$$

$i \backslash t$	John_1	carried_2	a_3	tin_4	can_5
N	0.005				
M	0				
V	0				
D	0				

final

The probability of the most probable sequence up to t ending with a tag i

Viterbi Algorithm

a_{ij}	STOP	N	M	V	D
START	0	0.5	0	0.1	0.4
N	0.2	0.2	0.3	0.3	0
M	0	0	0	1	0
V	0.1	0.3	0.3	0	0.3
D	0	1	0	0	0

b_{ik}	tin	John	can	carried	a	...
N	0.05	0.01	0.01	0	0	...
M	0	0	0.2	0	0	...
V	0	0	0	0.02	0	...
D	0	0	0	0	0.5	...

Init: $v_i^1 = a_{START,i} b_{i,x^1}, \quad i = 1, \ldots, N;$

Recursion: $v_j^t = \left(\max_i v_i^{t-1} a_{ij}\right) b_{j,x^t}, \; j = 1, \ldots, N, \; t = 2, \ldots, |x| - 1$

i \ t	John_1	carried_2	a_3	tin_4	can_5
N	0.005				
M	0				
V	0				
D	0				

final

The probability of the most probable sequence up to *t* ending with a tag *i*

Viterbi Algorithm

a_{ij}	STOP	N	M	V	D
START	0	0.5	0	0.1	0.4
N	0.2	0.2	0.3	$\boxed{0.3}$	0
M	0	0	0	1	0
V	0.1	0.3	0.3	0	0.3
D	0	1	0	0	0

b_{ik}	tin	John	can	carried	a	...
N	0.05	0.01	0.01	0	0	...
M	0	0	0.2	0	0	...
V	0	0	0	$\boxed{0.02}$	0	...
D	0	0	0	0	0.5	...

Init: $v_i^1 = a_{START,i} b_{i,x^1}, \quad i = 1, \ldots, N;$

Recursion: $v_j^t = \left(\max_i v_i^{t-1} a_{ij}\right) b_{j,x^t}, \ j = 1, \ldots, N, \ t = 2, \ldots, |x| - 1$

i \ t	John$_1$	carried$_2$	a$_3$	tin$_4$	can$_5$
N	0.005	0			
M	0	0			
V	0	$3 * 10^{-5}$			
D	0	0			

final

The probability of the most probable sequence up to t ending with a tag i

Viterbi Algorithm

a_{ij}	STOP	N	M	V	D
START	0	0.5	0	0.1	0.4
N	0.2	0.2	0.3	0.3	0
M	0	0	0	1	0
V	0.1	0.3	0.3	0	0.3
D	0	1	0	0	0

b_{ik}	tin	John	can	carried	a	...
N	0.05	0.01	0.01	0	0	...
M	0	0	0.2	0	0	...
V	0	0	0	0.02	0	...
D	0	0	0	0	0.5	...

Init: $$v_i^1 = a_{START,i} b_{i,x^1}, \quad i = 1, \ldots, N;$$

Recursion: $$v_j^t = \left(\max_i v_i^{t-1} a_{ij}\right) b_{j,x^t}, \; j = 1, \ldots, N, \; t = 2, \ldots, |x| - 1$$

i \ t	John_1	carried_2	a_3	tin_4	can_5
N	0.005	0	0		
M	0	0	0		
V	0	$3 * 10^{-5}$	0		
D	0	0	$4.5 * 10^{-5}$		

final

The probability of the most probable sequence up to t ending with a tag i

Viterbi Algorithm

上海科技大学 ShanghaiTech University

a_{ij}	STOP	N	M	V	D
START	0	0.5	0	0.1	0.4
N	0.2	0.2	0.3	0.3	0
M	0	0	0	1	0
V	0.1	0.3	0.3	0	0.3
D	0	$\boxed{1}$	0	0	0

b_{ik}	tin	John	can	carried	a	...
N	$\boxed{0.05}$	0.01	0.01	0	0	...
M	0	0	0.2	0	0	...
V	0	0	0	0.02	0	...
D	0	0	0	0	0.5	...

Init: $v_i^1 = a_{START,i} b_{i,x^1}, \quad i = 1, \ldots, N;$

Recursion: $v_j^t = \left(\max_i v_i^{t-1} a_{ij} \right) b_{j,x^t}, \; j = 1, \ldots, N, \; t = 2, \ldots, |x| - 1$

$i \backslash t$	John_1	carried_2	a_3	tin_4	can_5
N	0.005	0	0	$2.25 * 10^{-7}$	
M	0	0	0	0	
V	0	$3 * 10^{-5}$	0	0	
D	0	0	$4.5 * 10^{-5}$	0	

final

The probability of the most probable sequence up to t ending with a tag i

志成才 报国裕民

Viterbi Algorithm

a_{ij}	STOP	N	M	V	D
START	0	0.5	0	0.1	0.4
N	0.2	$\boxed{0.2}$	0.3	0.3	0
M	0	0	0	1	0
V	0.1	0.3	0.3	0	0.3
D	0	1	0	0	0

b_{ik}	tin	John	can	carried	a	...
N	0.05	0.01	$\boxed{0.01}$	0	0	...
M	0	0	0.2	0	0	...
V	0	0	0	0.02	0	...
D	0	0	0	0	0.5	...

Init: $v_i^1 = a_{START,i} b_{i,x^1}, \quad i = 1, \ldots, N;$

Recursion: $v_j^t = \left(\max_i v_i^{t-1} a_{ij}\right) b_{j,x^t}, \; j = 1, \ldots, N, \; t = 2, \ldots, |x| - 1$

$i \backslash t$	John_1	carried_2	a_3	tin_4	can_5
N	0.005	0	0	$2.25 * 10^{-7}$	$4.5 * 10^{-10}$
M	0	0	0	0	
V	0	$3 * 10^{-5}$	0	0	
D	0	0	$4.5 * 10^{-5}$	0	

final

The probability of the most probable sequence up to *t* ending with a tag *i*

Viterbi Algorithm

a_{ij}	STOP	N	M	V	D
START	0	0.5	0	0.1	0.4
N	0.2	0.2	$\boxed{0.3}$	0.3	0
M	0	0	0	1	0
V	0.1	0.3	0.3	0	0.3
D	0	1	0	0	0

b_{ik}	tin	John	can	carried	a	...
N	0.05	0.01	0.01	0	0	...
M	0	0	$\boxed{0.2}$	0	0	...
V	0	0	0	0.02	0	...
D	0	0	0	0	0.5	...

Init: $v_i^1 = a_{START,i} b_{i,x^1}, \quad i = 1, \ldots, N;$

Recursion: $v_j^t = \left(\max_i v_i^{t-1} a_{ij}\right) b_{j,x^t}, \; j = 1, \ldots, N, \; t = 2, \ldots, |x| - 1$

i \ t	John_1	$\mathrm{carried}_2$	a_3	tin_4	can_5
N	0.005	0	0	$2.25 * 10^{-7}$	$4.5 * 10^{-10}$
M	0	0	0	0	$1.35 * 10^{-8}$
V	0	$3 * 10^{-5}$	0	0	
D	0	0	$4.5 * 10^{-5}$	0	

final

The probability of the most probable sequence up to *t* ending with a tag *i*

Viterbi Algorithm

a_{ij}	STOP	N	M	V	D
START	0	0.5	0	0.1	0.4
N	0.2	0.2	0.3	0.3	0
M	0	0	0	1	0
V	0.1	0.3	0.3	0	0.3
D	0	1	0	0	0

b_{ik}	tin	John	can	carried	a	...
N	0.05	0.01	0.01	0	0	...
M	0	0	0.2	0	0	...
V	0	0	0	0.02	0	...
D	0	0	0	0	0.5	...

Init: $$v_i^1 = a_{START,i} b_{i,x^1}, \quad i = 1, \ldots, N;$$

Recursion: $$v_j^t = \left(\max_i v_i^{t-1} a_{ij}\right) b_{j,x^t}, \;\; j = 1, \ldots, N, \;\; t = 2, \ldots, |x| - 1$$

i \ t	John_1	carried_2	a_3	tin_4	can_5
N	0.005	0	0	$2.25 * 10^{-7}$	$4.5 * 10^{-10}$
M	0	0	0	0	$1.35 * 10^{-8}$
V	0	$3 * 10^{-5}$	0	0	0
D	0	0	$4.5 * 10^{-5}$	0	0

final

The probability of the most probable sequence up to *t* ending with a tag *i*

Viterbi Algorithm

a_{ij}	STOP	N	M	V	D
START	0	0.5	0	0.1	0.4
N	0.2	0.2	0.3	0.3	0
M	0	0	0	1	0
V	0.1	0.3	0.3	0	0.3
D	0	1	0	0	0

b_{ik}	tin	John	can	carried	a	...
N	0.05	0.01	0.01	0	0	...
M	0	0	0.2	0	0	...
V	0	0	0	0.02	0	...
D	0	0	0	0	0.5	...

Init: $v_i^1 = a_{START,i} b_{i,x^1}, \quad i = 1, \ldots, N;$

Recursion: $v_j^t = \left(\max_i v_i^{t-1} a_{ij}\right) b_{j,x^t}, \; j = 1, \ldots, N, \; t = 2, \ldots, |x| - 1$

Final: $v_{STOP}^{|\mathbf{x}|+1} = \max_i v_i^{|\mathbf{x}|} a_{i,STOP}$

$i \backslash t$	John_1	carried_2	a_3	tin_4	can_5
N	0.005	0	0	$2.25 * 10^{-7}$	$4.5 * 10^{-10}$
M	0	0	0	0	$1.35 * 10^{-8}$
V	0	$3 * 10^{-5}$	0	0	0
D	0	0	$4.5 * 10^{-5}$	0	0

final

At the final step, select the path with the highest probability

The probability of the most probable sequence up to t ending with a tag i

Viterbi Algorithm

a_{ij}	STOP	N	M	V	D
START	0	0.5	0	0.1	0.4
N	$\boxed{0.2}$	0.2	0.3	0.3	0
M	$\boxed{0}$	0	0	1	0
V	0.1	0.3	0.3	0	0.3
D	0	1	0	0	0

b_{ik}	tin	John	can	carried	a	...
N	0.05	0.01	0.01	0	0	...
M	0	0	0.2	0	0	...
V	0	0	0	0.02	0	...
D	0	0	0	0	0.5	...

Init: $v_i^1 = a_{START,i} b_{i,x^1}, \quad i = 1, \ldots, N;$

Recursion: $v_j^t = \left(\max_i v_i^{t-1} a_{ij}\right) b_{j,x^t}, \; j = 1, \ldots, N, \; t = 2, \ldots, |x| - 1$

Final: $v_{STOP}^{|\mathbf{x}|+1} = \max_i v_i^{|\mathbf{x}|} a_{i,STOP}$

At the final step, select the path with the highest probability

$i \backslash t$	John_1	carried_2	a_3	tin_4	can_5
N	0.005	0	0	$2.25 * 10^{-7}$	$4.5 * 10^{-10}$
M	0	0	0	0	$1.35 * 10^{-8}$
V	0	$3 * 10^{-5}$	0	0	0
D	0	0	$4.5 * 10^{-5}$	0	0

final

The probability of the most probable sequence up to *t* ending with a tag *i*

Viterbi Algorithm

a_{ij}	STOP	N	M	V	D
START	0	0.5	0	0.1	0.4
N	$\boxed{0.2}$	0.2	0.3	0.3	0
M	0	0	0	1	0
V	0.1	0.3	0.3	0	0.3
D	0	1	0	0	0

b_{ik}	tin	John	can	carried	a	...
N	0.05	0.01	0.01	0	0	...
M	0	0	0.2	0	0	...
V	0	0	0	0.02	0	...
D	0	0	0	0	0.5	...

Init: $v_i^1 = a_{START,i} b_{i,x^1}, \quad i = 1, \dots, N;$

Recursion: $v_j^t = \left(\max_i v_i^{t-1} a_{ij}\right) b_{j,x^t}, \; j = 1, \dots, N, \; t = 2, \dots, |x| - 1$

Final: $v_{STOP}^{|\mathbf{x}|+1} = \max_i v_i^{|\mathbf{x}|} a_{i,STOP}$

At the final step, select the path with the highest probability

$i \backslash t$	John_1	carried_2	a_3	tin_4	can_5
N	0.005	0	0	$2.25 * 10^{-7}$	$4.5 * 10^{-10}$
M	0	0	0	0	$1.35 * 10^{-8}$
V	0	$3 * 10^{-5}$	0	0	0
D	0	0	$4.5 * 10^{-5}$	0	0

final
$9 * 10^{-11}$

The probability of the most probable sequence up to t ending with a tag i

Viterbi Algorithm

	arried	a	...
ST	0	0	...
	0	0	...
	.02	0	...
	0	0.5	...

In

Recu

Fina

John carried a tin can
N V D N N

At the final step, select the path with the highest probability

$i \backslash t$	John_1	carried_2	a_3	tin_4	can_5
N	0.005	0	0	$2.25 * 10^{-7}$	$4.5 * 10^{-10}$
M	0	0	0	0	$1.35 * 10^{-8}$
V	0	$3 * 10^{-5}$	0	0	0
D	0	0	$4.5 * 10^{-5}$	0	0

final
$9 * 10^{-11}$

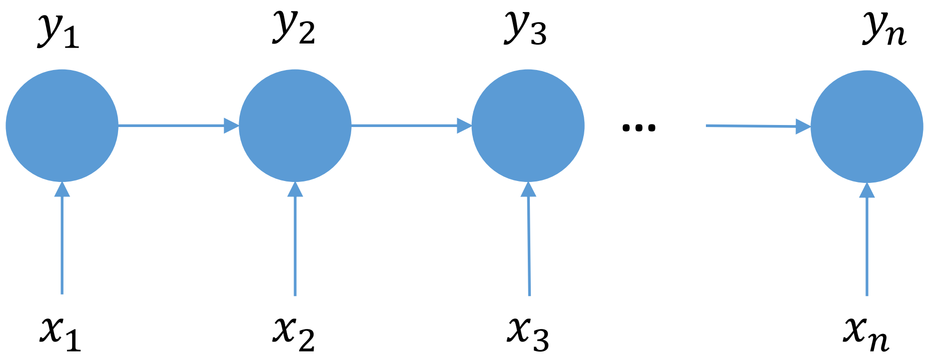
The probability of the most probable sequence up to t ending with a tag i

志成才 报国裕民

Beyond HMM

- The simplest method: for each word, predict its most frequent label
  - Problems:
    1. It does not consider the contextual info
    2. It does not consider relations between adjacent labels
- HMM handles problem 2, but not 1

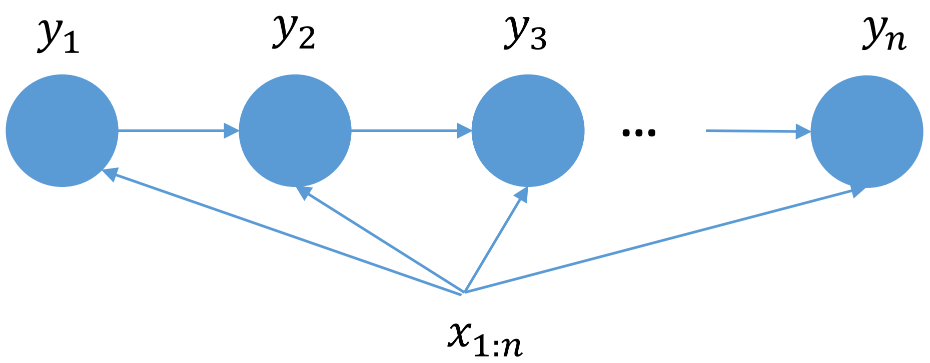
Max-Entropy Markov Models (MEMM)



$$P(y_{1:n}|x_{1:n}, W) = P(y_1|x_1, W) \prod_{t=1}^{n} P(y_t|y_{t-1}, x_t, W)$$

$$P(y_t|y_{t-1}, x_t, W) = \frac{\exp(W^T f(y_{t-1}, y_t, x_t))}{Z(y_{t-1}, x_t)}$$

Max-Entropy Markov Models (MEMM)

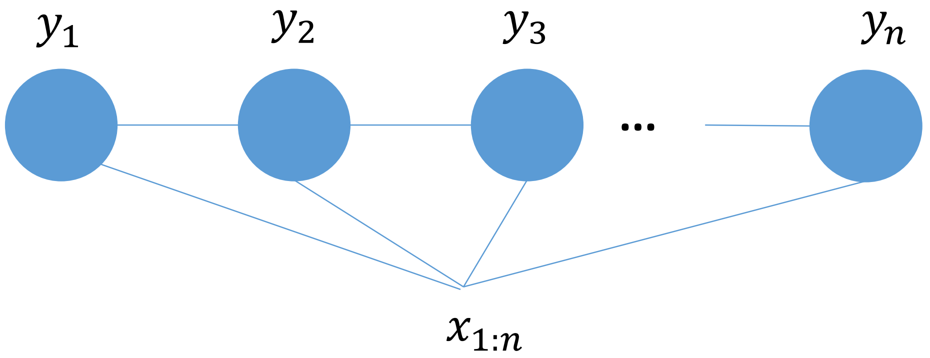


$$P(y_{1:n}|x_{1:n}, W) = P(y_1|x_{1:n}, W) \prod_{t=1}^{n} P(y_t|y_{t-1}, x_{1:n}, W)$$

$$P(y_t|y_{t-1}, x_{1:n}, W) = \frac{\exp(W^T f(y_{t-1}, y_t, x_{1:n}))}{Z(y_{t-1}, x_{1:n})}$$

- MEMM considers both contextual info and relations between adjacent labels!
- But… MEMM suffers from label bias problem

Conditional Random Field (CRF)



$$P(y_{1:n}|x_{1:n}, W) = \frac{1}{Z(x_{1:n}, W)} \prod_{t=1}^{n} \exp(W^T f(y_{t-1}, y_t, x_{1:n}))$$

- CRF is an undirected graphical model
  - Global normalization instead of local normalization

CRF Inference

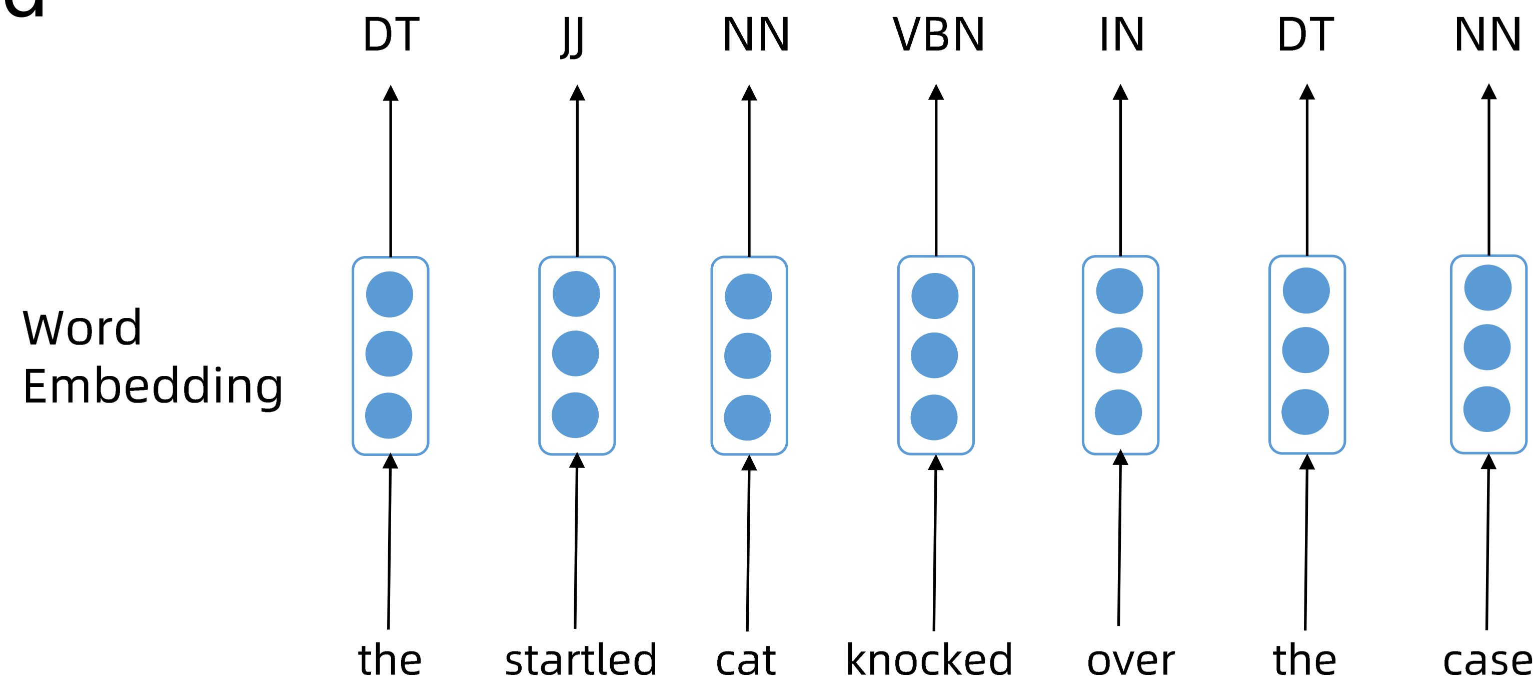
- Find the most likely label sequence of a sentence
- Inference objective

$$y_{1:n}{}^{*} = \arg\max_{y_{1:n}} \exp\left(\sum_{t=1}^{n} W^T f(y_{t-1}, y_t, x_{1:n})\right)$$

- Inference Algorithm
  - Viterbi

Neural Methods

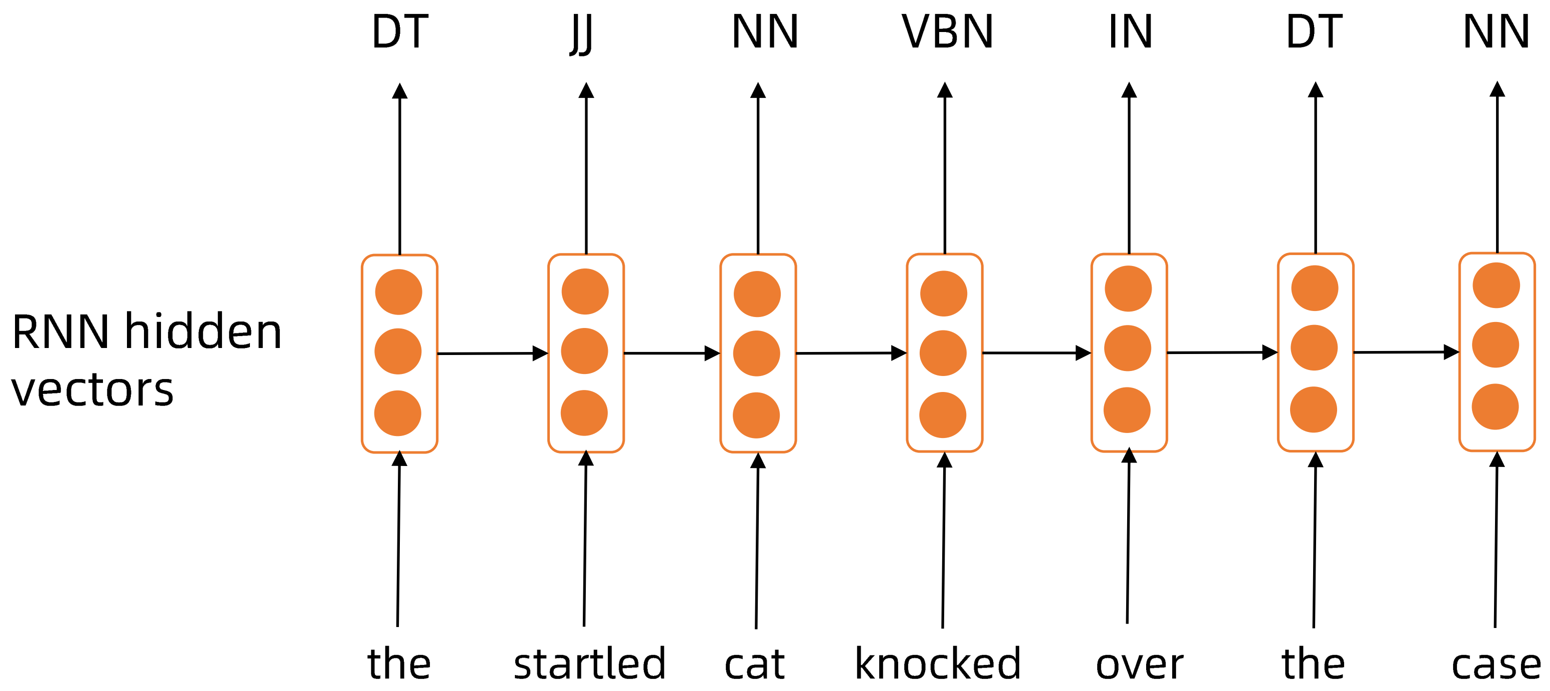
- Simplest NN: predict a label from the embedding of each word



1. It does not consider the contextual info ☹
2. It does not consider relations between adjacent labels ☹

Neural Methods

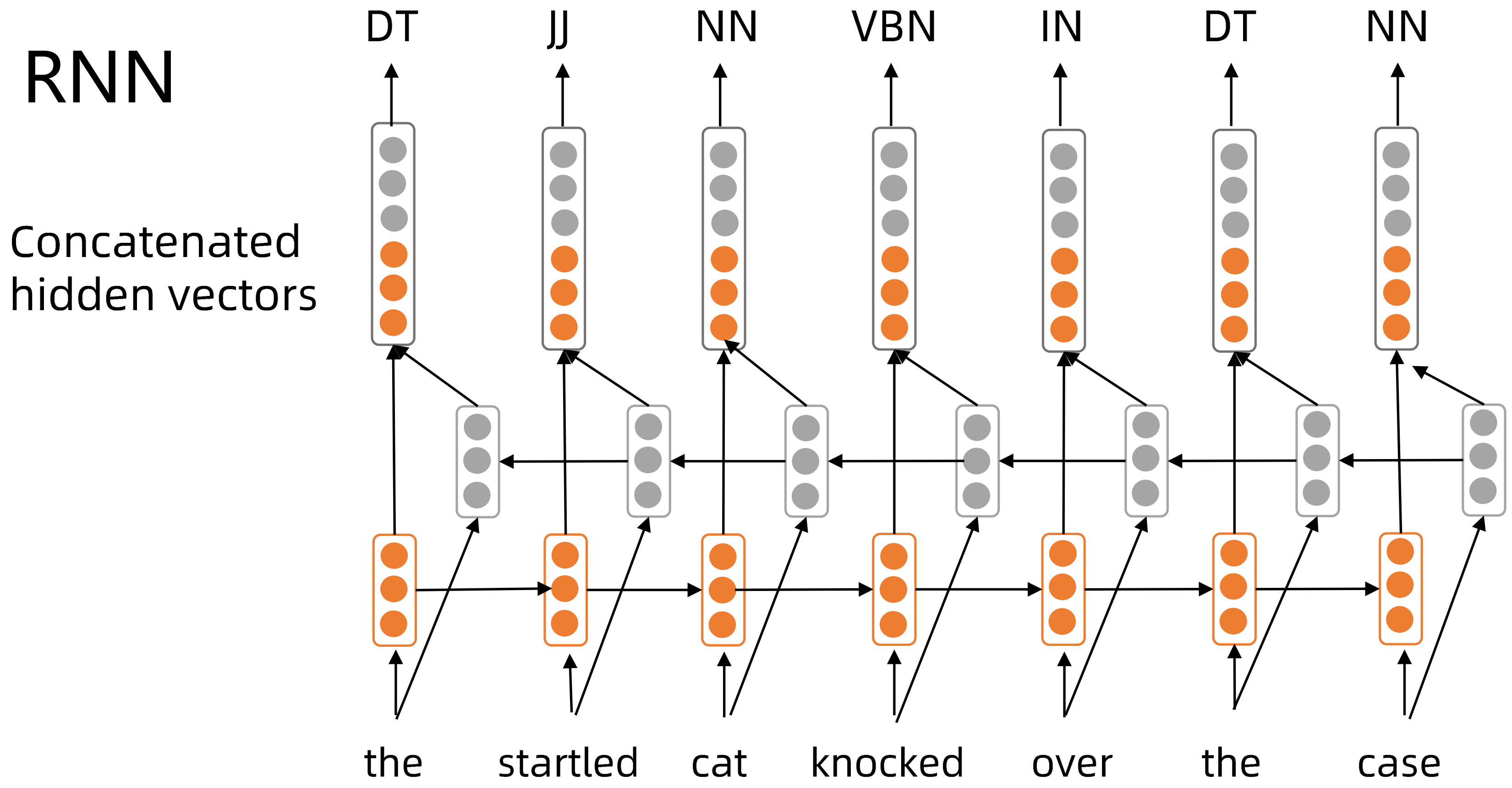
- Recurrent neural network (RNN)



1. It does not consider the contextual info ☹ ← Only the left context of each word is used
2. It does not consider relations between adjacent labels ☹

Neural Methods

- Bidirectional RNN

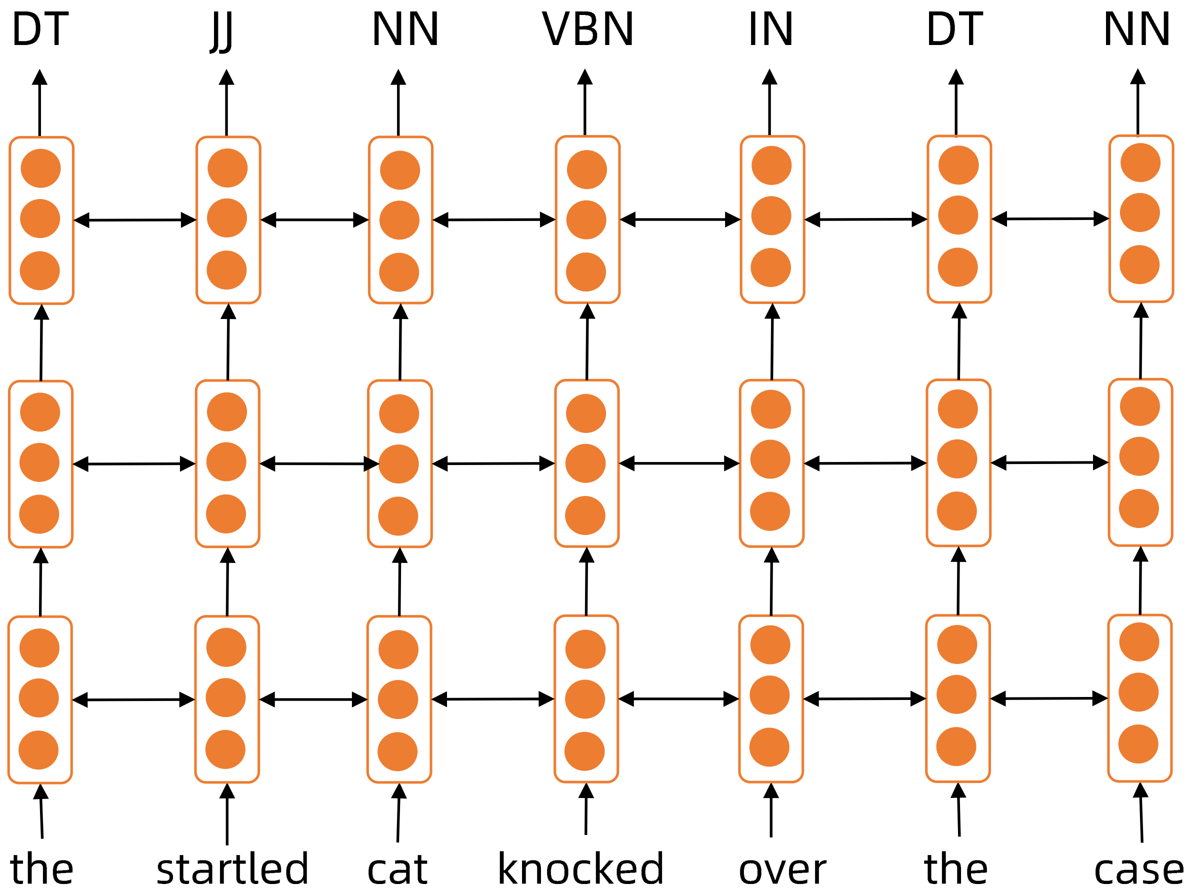


1. ~~It does not consider the contextual info~~
2. It does not consider relations between adjacent labels

Neural Methods

上海科技大学 ShanghaiTech University

- Multilayer Bidirectional RNN

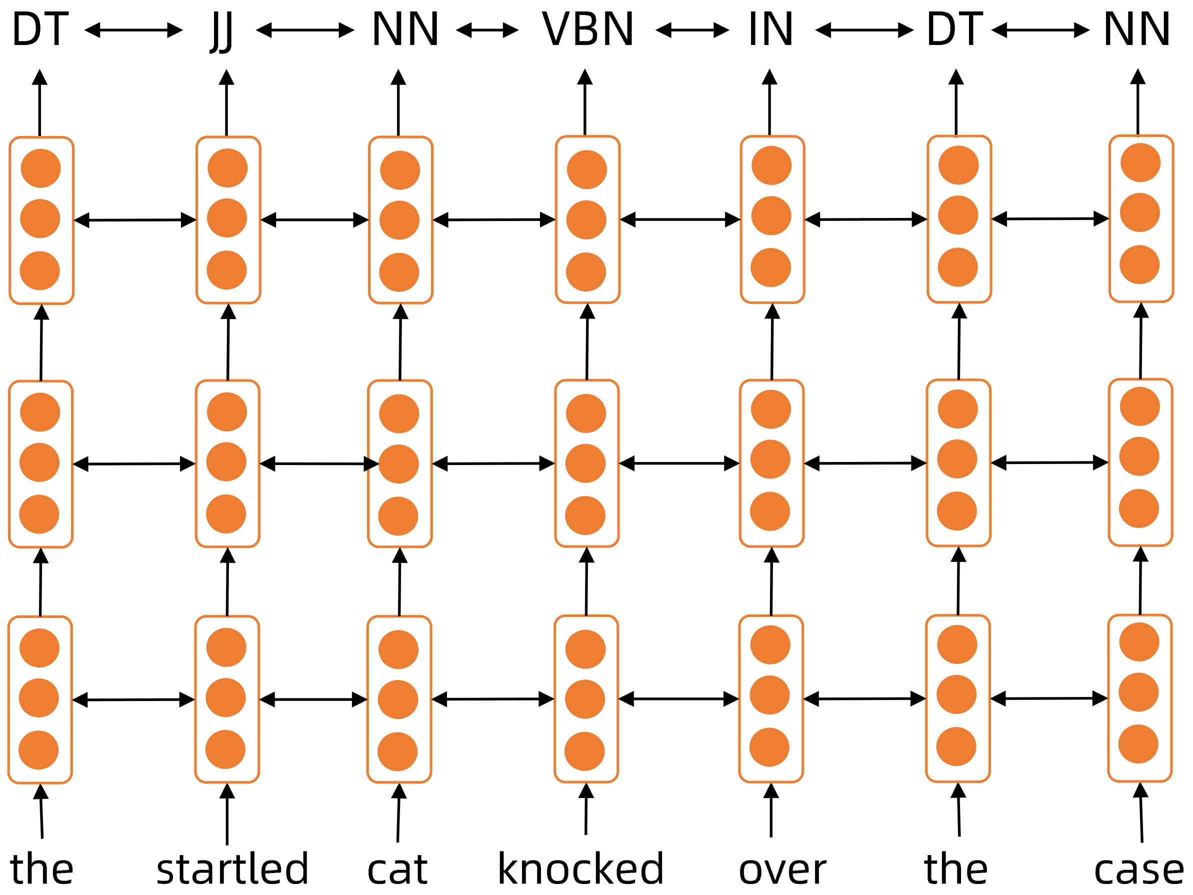


~~1. It does not consider the contextual info~~ Captures more contextual info

2. It does not consider relations between adjacent labels

Neural Methods

- Multilayer Bidirectional RNN +CRF decoder



1. ~~It does not consider the contextual info~~ 😆
2. ~~It does not consider relations between adjacent labels~~ 😆

立志 成才 报国 裕民

Summary

- Sequence labeling
  - Predict a label for each word of a sentence
  - Many NLP tasks can be seen as sequence labeling
- Methods
  - HMM
  - MEMM
  - CRF
  - RNN

Knowledge Representation

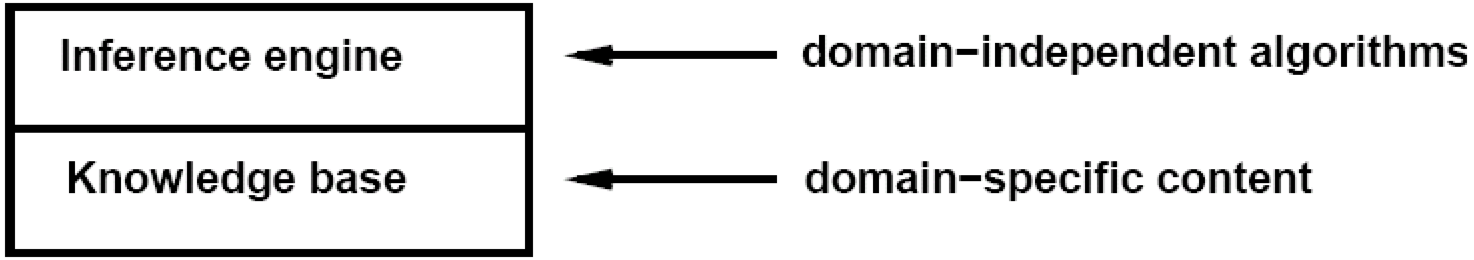
Knowledge Representation (KR)

- Representation
  - How to represent information in a form that a computer can utilize to solve tasks
- Reasoning
  - How to utilize represented knowledge to derive new knowledge

- Majority of KR research is based on symbolic logic

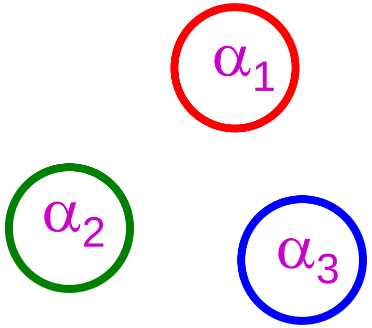
Logic

- Logic
  - Formal language in which knowledge can be expressed
  - A means of carrying out reasoning in the language
- Logic AI
  - Knowledge base: set of sentences in a formal language to represent knowledge about the "world"
  - Inference engine: answers any answerable question following the knowledge base

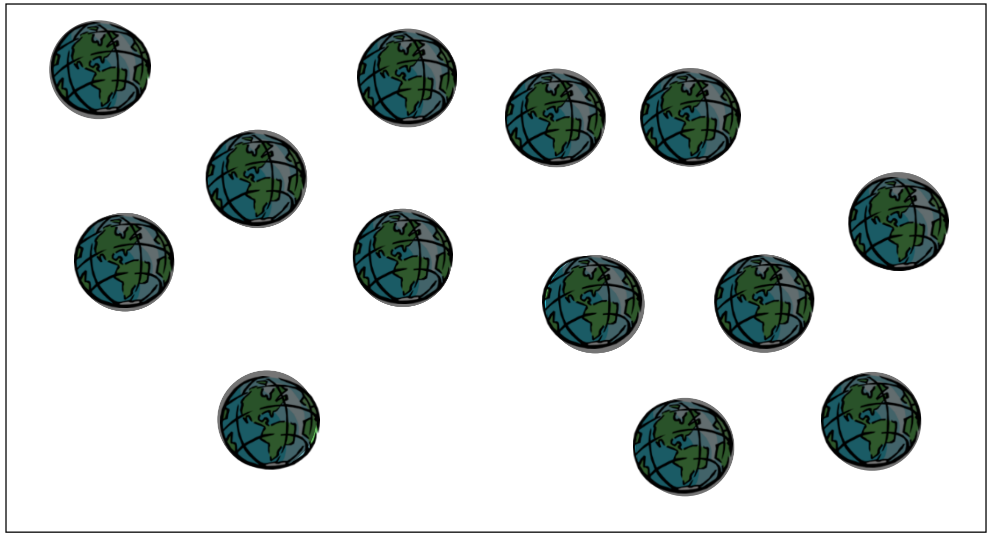


Formal Language

- Components of a formal language in a logic
  - Syntax: What sentences are allowed?
  - Semantics:
    - Which sentences are true/false in each model (possible world)?



Syntaxland



Semanticsland

Formal Language

- Example: the language of arithmetic
  - Syntax
    - $x+2 \geq y$ is a sentence
    - x2+y > {} is not a sentence
  - Semantics
    - $x+2 \geq y$ is true in a world where $x = 7, y = 1$
    - $x+2 \geq y$ is false in a world where $x = 0, y = 6$

KR – Propositional logic

Propositional logic: Syntax

上海科技大学
ShanghaiTech University

- Propositional logic is the "simplest" logic
  - The proposition symbols P1, P2, etc. are sentences
  - If S is a sentence, $\neg$S is a sentence (negation)
  - If S1 and S2 are sentences, S1 $\wedge$ S2 is a sentence (conjunction)
  - If S1 and S2 are sentences, S1 $\vee$ S2 is a sentence (disjunction)
  - If S1 and S2 are sentences, S1 $\Rightarrow$ S2 is a sentence (implication)
  - If S1 and S2 are sentences, S1 $\Leftrightarrow$ S2 is a sentence (biconditional)

$\neg$, $\wedge$, $\vee$, $\Rightarrow$, $\Leftrightarrow$ are called logic connectives or operators

Sometimes $\rightarrow$ and $\leftrightarrow$ are used

Examples of PL sentences

- P means "It is hot."
- Q means "It is humid."
- R means "It is raining."
- $(P \wedge Q) \Rightarrow R$
  - "If it is hot and humid, then it is raining"
- $Q \Rightarrow P$
  - "If it is humid, then it is hot"

Propositional logic: Semantics

- Each model specifies true/false for each proposition symbol
  - E.g.

	P1	P2	P3
	false	true	false

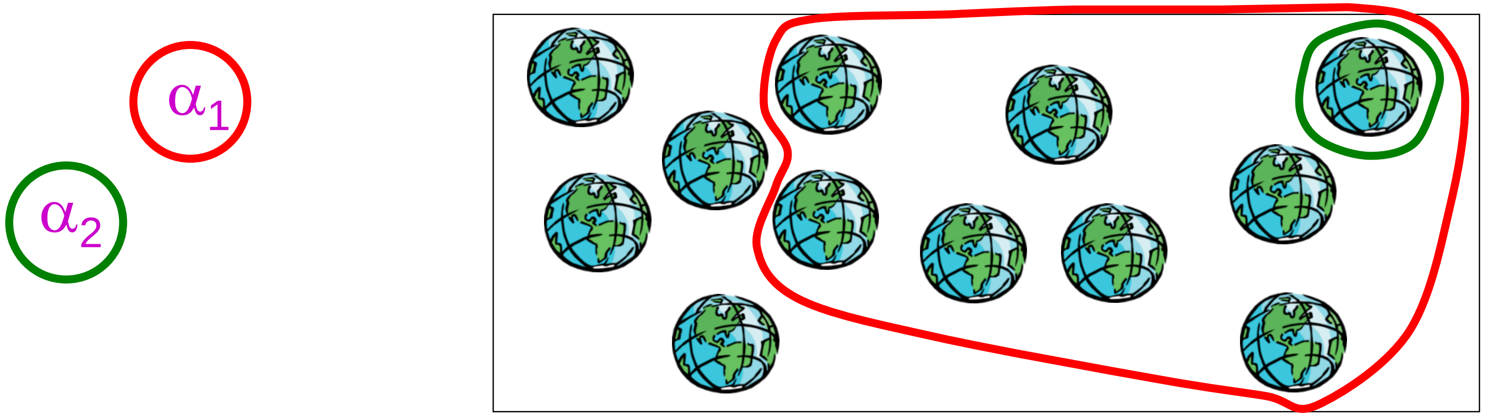
- Rules for evaluating truth with respect to a model m:
  - $\neg$S is true iff S is false
  - S1 $\wedge$ S2 is true iff S1 is true and S2 is true
  - S1 $\vee$ S2 is true iff S1 is true or S2 is true
  - S1 $\Rightarrow$ S2 is true iff S1 is false or S2 is true
  - S1 $\Leftrightarrow$ S2 is true iff S1$\Rightarrow$S2 is true and S2$\Rightarrow$S1 is true

Truth tables for connectives

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
$false$	$false$	$true$	$false$	$false$	$true$	$true$
$false$	$true$	$true$	$false$	$true$	$true$	$false$
$true$	$false$	$false$	$false$	$true$	$false$	$false$
$true$	$true$	$false$	$true$	$true$	$true$	$true$

Inference: entailment

- Entailment: $\alpha \models \beta$ ("α entails β" or "β follows from α") means in every world where α is true, β is also true
  - i.e., the α-worlds are a subset of the β-worlds [models(α) $\subseteq$ models(β)]
- In the example, $\alpha 2 \models \alpha 1$



Inference: proof

- A proof (α |- β) is a demonstration of entailment from α to β
- Method: application of inference rules
  - Search for a finite sequence of sentences each of which is an axiom or follows from the preceding sentences by a rule of inference
  - Axiom: a sentence known to be true
  - Rule of inference: a function that takes one or more sentences (premises) and returns a sentence (conclusion)

Inference: soundness & completeness

- Sound inference
  - everything that can be proved is in fact entailed
- Complete inference
  - everything that is entailed can be proved

- Almost every logic that we use is sound
- Not every logic is complete
  - Example: arithmetic is found to be not complete! (Gödel's theorem, 1931)

Resolution: an inference rule in PL

- Conjunctive Normal Form (CNF)
  - conjunction of disjunctions of literals (clauses)
  - Ex

$(A \vee \neg B) \wedge (B \vee \neg C \vee \neg D)$

$(\neg B1,1 \vee P1,2 \vee P2,1) \wedge (\neg P1,2 \vee B1,1) \wedge (\neg P2,1 \vee B1,1)$

Resolution: an inference rule in PL

上海科技大学 ShanghaiTech University

- Resolution inference rule (for CNF):
  Suppose l_i is $\neg m_j$

$$\frac{l_1 \vee \ldots \vee l_k, \qquad m_1 \vee \ldots \vee m_n}{l_1 \vee \ldots \vee l_{i-1} \vee l_{i+1} \vee \ldots \vee l_k \vee m_1 \vee \ldots \vee m_{j-1} \vee m_{j+1} \vee \ldots \vee m_n}$$

Examples:

$$\frac{P_{1,3} \vee P_{2,2}, \qquad P_{2,3} \vee \neg P_{2,2}}{P_{1,3} \vee P_{2,3}} \qquad \frac{P_1, \neg P_1}{\{\}}$$

- Resolution is sound and complete for propositional logic

立志成才报国裕民

Resolution algorithm

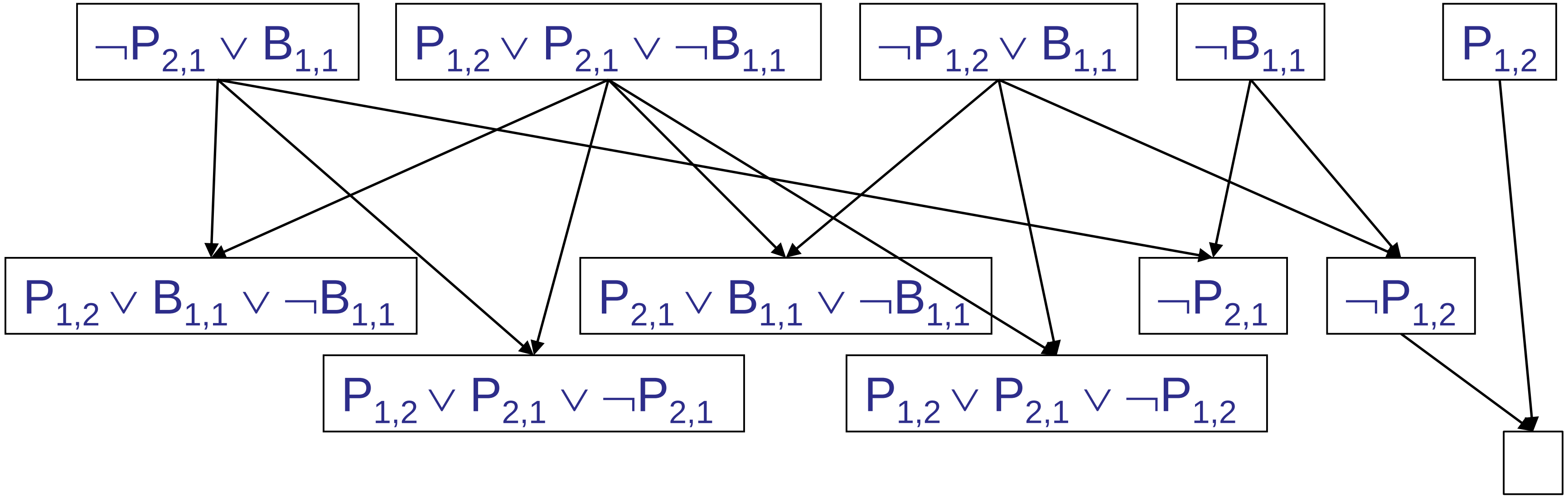
- The best way to prove $KB \models \alpha$?
  - Proof by contradiction, i.e., show $KB \wedge \neg\alpha$ is unsatisfiable

  1. Convert $KB \wedge \neg\alpha$ to CNF
  2. Repeatedly apply the resolution rule to add new clauses, until one of the two things happens
     a) Two clauses resolve to yield the empty clause, in which case KB entails α
     b) There is no new clause that can be added, in which case KB does not entail α

Resolution example

$KB = (B_{1,1} \Leftrightarrow (P_{1,2} \lor P_{2,1})) \land \neg B_{1,1}$

$\alpha = \neg P_{1,2}$



KR – First-order predicate logic

Syntax of FOL: Basic elements

- Logical symbols
  - Connectives $\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow$
  - Quantifiers $\forall, \exists$
  - Variables x, y, a, b, ...
  - Equality =
- Non-logical symbols (ontology)
  - Constants KingArthur, 2, ShanghaiTech, ...
  - Predicates Brother, >, ...
  - Functions Sqrt, LeftLegOf, ...

Atomic sentences

Atomic sentence = *predicate* (*term*$_1$,...,*term*$_n$)
or *term*$_1$ = *term*$_2$

Term = *constant* or *variable*
or *function* (*term*$_1$,...,*term*$_n$)

Example:

Brother(KingJohn,RichardTheLionheart)

>(Length(LeftLegOf(Richard)), Length(LeftLegOf(KingJohn)))

Complex sentences

Complex sentences are made from atomic sentences using connectives

$$\neg S, \quad S_1 \land S_2, \quad S_1 \lor S_2, \quad S_1 \Rightarrow S_2, \quad S_1 \Leftrightarrow S_2,$$

Example:

$Sibling(KingJohn,Richard) \Rightarrow Sibling(Richard,KingJohn)$

$>(1,2) \lor \leq (1,2)$

$>(1,2) \land \neg >(1,2)$

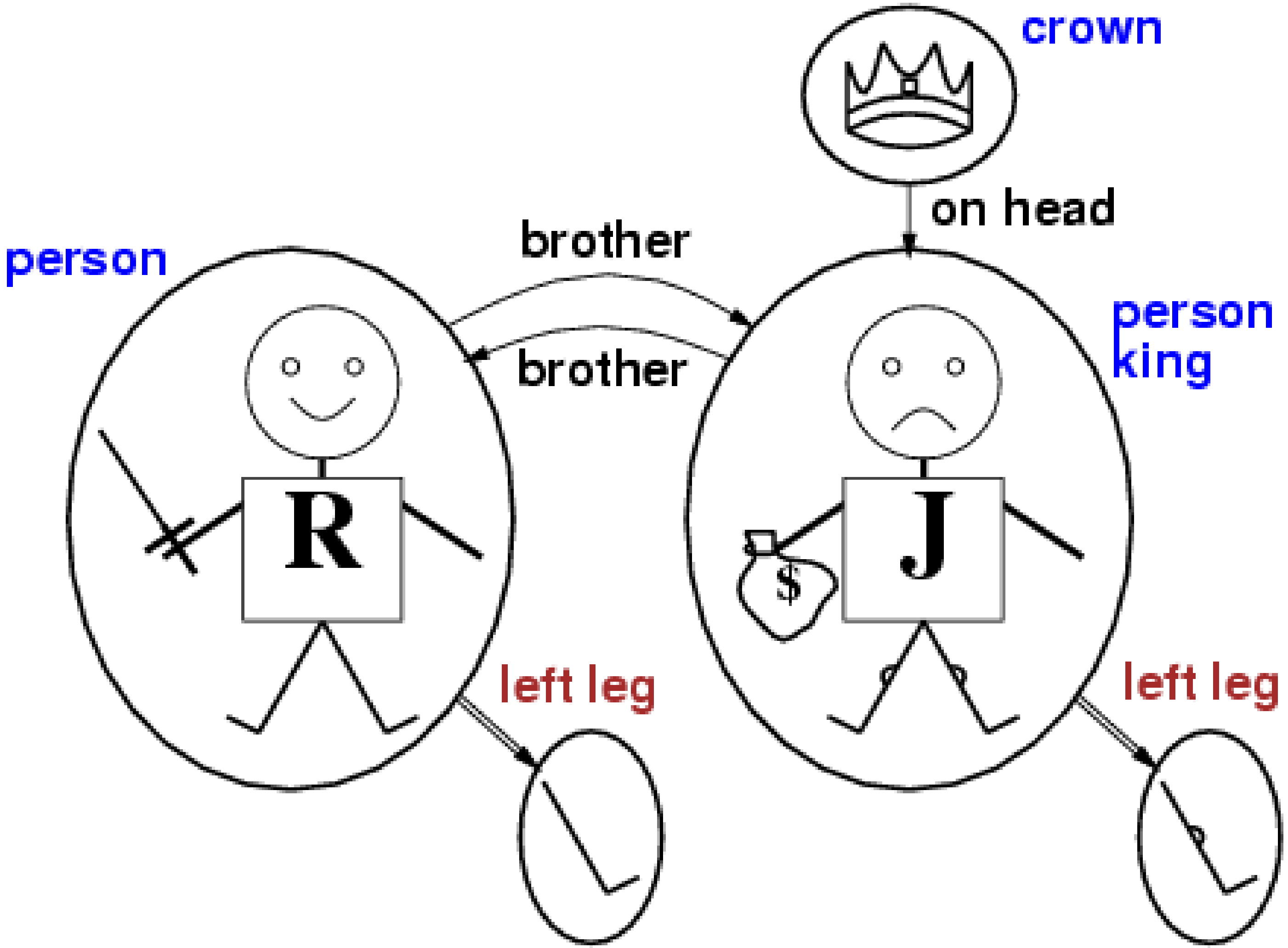
Semantics of FOL

- Sentences are true with respect to a model, which contains
  - Objects and relations among them
  - Interpretation specifying referents for

constant symbols	→	objects
predicate symbols	→	relations
function symbols	→	functional relations

- An atomic sentence *predicate*$(term_1,\ldots,term_n)$ is true iff the objects referred to by $term_1,\ldots,term_n$ are in the relation referred to by *predicate*

Models for FOL: Example



Models for FOL: Example

- Consider the interpretation:

  Richard → Person R
  John → Person J
  Brother → the brotherhood relation

  Under this interpretation, *Brother*(*Richard*, *John*) is true in the model.

Quantifiers

上海科技大学 ShanghaiTech University

- Allows us to express properties of collections of objects instead of enumerating objects by name
- Universal: “for all” $\forall$
- Existential: “there exists” $\exists$

立志成才报国裕民

Universal quantification

$\forall$<variables> <sentence>

Example: $\forall x\ \ At(x,STU) \Rightarrow Smart(x)$

(Everyone at ShanghaiTech is smart)

$\forall x\ P$ is true in a model m iff P is true with x being each possible object in the model

- Roughly speaking, equivalent to the conjunction of instantiations of P

$$
\begin{aligned}
& At(John,STU) \Rightarrow Smart(John) \\
\wedge\ \ & At(Richard,STU) \Rightarrow Smart(Richard) \\
\wedge\ \ & At(STU,STU) \Rightarrow Smart(STU) \\
\wedge\ \ & \ldots
\end{aligned}
$$

立志 成才 报国 裕民

Existential quantification

$\exists$<variables> <sentence>

Example: $\exists x \;\; At(x, STU) \wedge Smart(x)$

(Someone at ShanghaiTech is smart)

$\exists x \, P$ is true in a model m iff P is true with x being some possible object in the model

- Roughly speaking, equivalent to the disjunction of instantiations of P

$$
\begin{aligned}
& (At(John, STU) \wedge Smart(John)) \\
\vee \;\; & (At(Richard, STU) \wedge Smart(Richard)) \\
\vee \;\; & (At(STU, STU) \wedge Smart(STU)) \\
\vee \;\; & \ldots
\end{aligned}
$$

FOL example: kinship

- Brothers are siblings
  $\forall x,y\ Brother(x,y) \Rightarrow Sibling(x,y).$
- "Sibling" is symmetric
  $\forall x,y\ Sibling(x,y) \Leftrightarrow Sibling(y,x).$
- One's mother is one's female parent
  $\forall x,y\ Mother(x,y) \Leftrightarrow (Female(x) \land Parent(x,y)).$
- A first cousin is a child of a parent's sibling
  $\forall x,y\ FirstCousin(x,y) \Leftrightarrow \exists p,ps\ Parent(p,x) \land Sibling(ps,p) \land Parent(ps,y)$

Resolution for FOL

- Full first-order version:

$$\frac{\ell_1 \vee \cdots \vee \ell_k, \qquad m_1 \vee \cdots \vee m_n}{(\ell_1 \vee \cdots \vee \ell_{i-1} \vee \ell_{i+1} \vee \cdots \vee \ell_k \vee m_1 \vee \cdots \vee m_{j-1} \vee m_{j+1} \vee \cdots \vee m_n)\theta}$$

where $\mathtt{Unify}(\ell_i, \neg m_j) = \theta$.

  - The two clauses are assumed to be standardized apart so that they share no variables.

- Example:

$$\frac{\neg Rich(x) \vee Unhappy(x) \qquad Rich(Ken)}{Unhappy(Ken)}$$

with $\theta = \{x/Ken\}$

- Inference algorithm: applying resolution steps to CNF(KB $\wedge \neg\alpha$)
- Resolution is sound and complete for FOL

Summary

- Knowledge Representation
  - Represent knowledge that computers can utilize
  - Reasoning about new knowledge
- Propositional Logic
  - Proposition symbols + logic connectives
- First-Order Predicate Logic
  - Constants, variables, predicates, quantifiers
  - Objects, relations