Sequence labeling: POS Tagger

Knowledge & Language Engineering Lab.

Contents

- Sequence Labeling
  - Introduction
  - Method
- Practice
  - Simple POS tagger using HMM Algorithm

[Relation Extraction]

SEQUENCE LABELING

Introduction

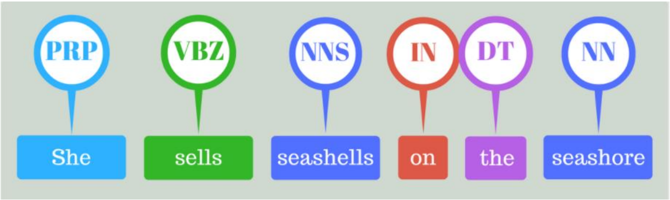
- Sequence labeling
  - A pattern recognition task that classifies a categorical label to each member of a sequence elements.
  - In NLP, which deals with sequential data, sequence labeling is one of the major task.

- Tasks or subtasks

Named entity recognition

Automatically find names of people, places, products, and organizations in text across many languages.

Part of speech tagging



Spacing problem

아버지가방에들어가신다.

↓

아버지가 방에 들어가신다.

Introduction

- Sequential Data
  - Data stored in chronological order.
  - Generally, each element is related to each other.
  - E.g.)
    - Video: a sequence of frames
    - Text: a sequence of words
    - Voice: a sequence of signals.

Methods

- Sequence labeling methods
  - Vector space model
    - Neural network model
    - Structured SVM
  - Probabilistic model
    - Hidden Markov Model (HMM)
    - Conditional Random Field (CRF)

Methods

- Sequence labeling methods
  - Vector space model
    - Neural network model
    - Structured SVM
  - Probabilistic model
    - **Hidden Markov Model (HMM)**
    - Conditional Random Field (CRF)

Hidden Markov Model

- $y^*_{1:N} = argmax_{y_{1:N}} P(y_{1:N}|x_{1:N})$

  $(Bayes\ rule)$

  $= argmax_{y_{1:N}} P(x_{1:N}|y_{1:N})P(y_{1:N})$

  $= argmax_{y_{1:N}} \prod_{k=1}^{N} P(x_k|x_{1:k-1}, y_{1:K}) \prod_{k=1}^{N} P(y_k|y_{1:k-1})$

  $(Markov\ assumption)$

  $\approx argmax_{y_{1:N}} \prod_{k=1}^{N} P(x_k|y_k) \prod_{k=1}^{N} P(y_k|y_{k-1})$

Hidden Markov Model

- $y^*_{1:N} = argmax_{y_{1:N}} P(y_{1:N}|x_{1:N})$

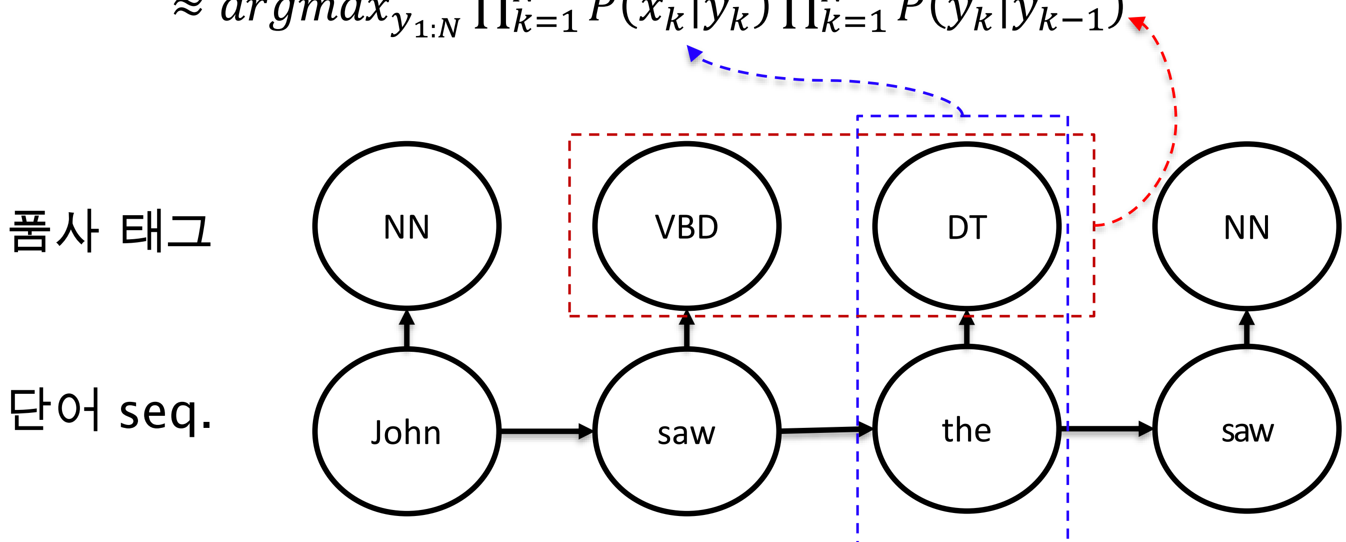
$(Bayes\ rule)$

$$= argmax_{y_{1:N}} P(x_{1:N}|y_{1:N})P(y_{1:N})$$

$$= argmax_{y_{1:N}} \prod_{k=1}^{N} P(x_k|x_{1:k-1}, y_{1:N}) \prod_{k=1}^{N} P(y_k|y_{1:k-1})$$

$(Markov\ assumption)$

$$\approx argmax_{y_{1:N}} \prod_{k=1}^{N} P(x_k|y_k) \prod_{k=1}^{N} P(y_k|y_{k-1})$$



Hidden Markov Model

- $argmax_{y_{1:N}} \prod_{k=1}^{N} P(x_k|y_k) \prod_{k=1}^{N} P(y_k|y_{k-1})$

  - $P(x_k|y_k)$: emission probability
    - 각 state(y) 에서 관측 가능한 값(x)의 확률
    - E.g.) 명사(NN) 인 ‘saw’ 가 등장할 확률
    - $P(x_k|y_k) = \frac{P(x_k,y_k)}{P(y_k)}$

  - $P(y_k|y_{k-1})$: transition probability
    - State(y) 간의 변화 확률
    - E.g.) 동사(VB) 이후에 명사(NN)가 등장할 확률
    - $P(y_k|y_{k-1}) = \frac{P(y_k,y_{k-1})}{P(y_{k-1})}$

Hidden Markov Model

- $\log(P(\text{NN VBD DT NN}|\text{John saw the saw }))$

$$= \log P(Jone|NN) + \log P(NN| < BOS >)$$

$$+ \log P(saw|VBD) + \log P(VBD|NN)$$

$$+ \log P(the|DT) + \log P(DT|VBD)$$

$$+ \log P(saw|NN) + \log P(NN|DT)$$

$$+ \log P(< EOS > |NN)$$

PRACTICE

KLE tagset

- 부가자료: KLE_Tagset.docx 파일 참고

preprocessing

- Preprocess each line with a list of tuples.
  - $[[(word_1, tag_1), (word_2, tag_2), \ldots, (word_n, tag_n),$
    $[(word_1, tag_1), (word_2, tag_2), \ldots, (word_n, tag_n)$
    $\vdots$
    $[(word_1, tag_1), (word_2, tag_2), \ldots, (word_n, tag_n)]]$

```
그/CT 도/fjb 강하/YBH ㄴ/fmotg 카리스마/CMC 를/fjco 필요/CMC 하/fph ㅂ니다/fmof ./g
애플/CMC 이/fjcs 80/CS %/g 로/fjcao 그/SG 뒤/CMC 를/fjco 쫓/YBD 았/fmb 습니다/fmof ./g
이제/SBO 참가자들/CMC 이/fjcs 기념촬영/CMC 을/fjco 하/YBD 고/fmoc 있/YA 다/fmof ./g
```

↓

```
[[(그, CT), (도, fjb), (강하, YBH), ..., (ㅂ니다, fmof), (., g)],
 [(애플, CMC), (이, fjcs), (80, CS), …, (습니다, fmof), (., g)],
 [(이제, SBO), (참가자들, CMC), (이, fjcs) ,…, (다, fmof), (., g)]]
```

Train function

- Count the number of (word, tag)
  - Nested dictionary type
    - pos2words_freq = defaultdict(lambda: defaultdict(int))
    - Pos2words[pos][word] _freq:
      - stores the number (frequency) of (word, tag)

- Count the number of bigram tags (tag_{i-1}, tag_i)
  - Dictionary type
    - Define trans_freq = defaultdict(int) for bigrams counts
    - Define bos_freq = defaultdict(int) for the bigrams counts containing "BOS"
      - Trans[(tag_{i-1}, tag_i)] stores the number of bigrams
      - Bos[tag_i] stores the number of BOS bigrams

Train function

- Example
  - pos2words_freq

```
{CMC: {아버지: 10, 올림픽: 15, ..},
 CMP: {구글: 20, 애플: 15, ..}
 YBD: {마시: 10, 듣: 20, ...}}
```

  - trans_freq

```
{(CMC, fjb): 20, (CMP, fjb): 31, (fjco, fd): 55, ..}
```

  - bos_freq

```
{CMP: 100, CMC: 200, CT: 55, ...}
```

Train function

- Frequency → probability
  - pos2words_prob

```
{CMC: {아버지: 0.1, 올림픽: 0.2, ..},
 CMP: {구글: 0.05, 애플: 0.03, ..}
 YBD: {마시: 0.1, 듣: 0.2, ...}}
```

sum = 1.0

  - trans_prob

```
{(CMC, fjb): 0.2, (CMP, fjb): 0.1, (fjco, fd): 0.48, ..}
```

  - bos_prob

```
{CMP: 0.3, CMC: 0.1, CT: 0.24, ...}
```

Train function

- Frequency → probability
  - pos2words_prob

```
{CMC: {아버지: 0.1, 올림픽: 0.2, ..},
 CMP: {구글: 0.05, 애플: 0.03, ..}
 YBD: {마시: 0.1, 든: 0.2, ...}}
```

sum = 1.0

$P(x_k = \text{애플} \mid y_k = \mathrm{CMP}) = 0.03$

  - trans_prob

```
{(CMC, fjb): 0.2, (CMP, fjb): 0.1, (fjco, fd): 0.48, ..}
```

  - bos_prob

```
{CMP: 0.3, CMC: 0.1, CT: 0.24, ...}
```

Train function

- Frequency → probability
  - pos2words_prob

```
{CMC: {아버지: 0.1, 올림픽: 0.2, ..},
 CMP: {구글: 0.05, 애플: 0.03, ..}
 YBD: {마시: 0.1, 든: 0.2, ...}}
```

sum = 1.0

$P(x_k = \text{애플} \mid y_k = \mathrm{CMP}) = 0.03$

  - trans_prob

```
{(CMC, fjb): 0.2, (CMP, fjb): 0.1, (fjco, fd): 0.48, ..}
```

$\mathrm{P}(\mathrm{y}_{\mathrm{k}-1} = \mathrm{fjco} \mid \mathrm{y}_{\mathrm{k}} = \mathrm{fd}) = 0.48$

  - bos_prob

```
{CMP: 0.3, CMC: 0.1, CT: 0.24, ...}
```

Train function

- Emission probability
  - $P(x_k|y_k) = \frac{P(x_k,y_k)}{P(y_k)} = \frac{\#\ of\ (word_k, tag_k)}{\#\ of\ tag_k}$
- Transition probability
  - $P(y_k|y_{k-1}) = \frac{P(y_k,y_{k-1})}{P(y_{k-1})} = \frac{\#\ of\ (tag_{k-1}, tag_k)}{\#\ of\ tag_{k-1}}$

Inference

- For given input sentences
  - "감기/CMC 는/fjb 줄이/YBD 다/fmof ./g“
  - "감기/fmotg 는/fjb 줄/CMC 이다/fjj ./g“

- Calculate the log probability
  - $\log(\prod_{k=1}^{N} P(x_k|y_k) \prod_{k=1}^{N} P(y_k|y_{k-1}))$

$$= \sum \log P(x_k|y_k) + \log P(y_k|y_{k-1})$$

- Results

```
감기/CMC 는/fjb 줄이/YBD 다/fmof ./g: -5.489636
감기/fmotg 는/fjb 줄/CMC 이다/fjj ./g: -14.037157
```

END