# Natural Language Processing


**Yue Zhang**
**Westlake University**



西湖大學
WESTLAKE UNIVERSITY

## Chapter 9

# Sequence Segmentation

# Contents

# Contents

# Sequence Segmentation Task

WestlakeNLP

- Input: a character/word sequence $X_{1:n}$
- Output: the most probable segment sequence $\widehat{S_{1:|S|}}$

| | | |
|---|---|---|
| Word segmentation | Input | 那几年,南京市里面和米很贵 |
| | Output | 那(Those) 几(few) 年(years) , 南京市(Nanjing City) 里(in) 面(flour) 和(and) 米(rice) 很(very) 贵(expensive) |
| | Labels | S S S S B M E S S S S S S |
| Syntactic chunking | Input | Mary went to Chicago to meet her boyfriend John Smith. |
| | Output | $[\text{Mary}]_{NP}$ $[\text{went}]_{VP}$ $[\text{to}]_{PP}$ $[\text{Chicago}]_{NP}$ $[\text{to}]_{PP}$ $[\text{meet}]_{VP}$ $[\text{her boyfriend John Smith}]_{NP}$. |
| | Labels | B-NP B-VP B-PP B-NP B-PP B-VP B-NP I-NP I-NP I-NP |
| Named entity recognition | Input | Mary went to Chicago to meet her boyfriend John Smith. |
| | Output | $[\text{Mary}]_{PER}$ went to $[\text{Chicago}]_{LOC}$ to meet her boyfriend $[\text{John Smith}]_{PER}$. |
| | Labels | B-PER O O B-LOC O O O O B-PER I-PER |

# Contents

# Evaluating sequence segmentation

- Represent the output of sequence segmentation
  - a set of tuples $\{(b_i, e_i, l_i)\}$
  - $b_i$, $e_i$ and $l_i$ represent the beginning index, end index and label (if applicable) of a segment
- Metrics

  Given a gold output $S_g$ and a system output $S$, we can find a common subset of segments $S_m = S_g \cap S$.
  - precision: $P = \frac{S_m}{S}$ : percentage of segments in $S$ that are correct
  - recall: $R = \frac{S_m}{S_g}$: percentage of gold segments that are predicted
  - F-score: $F = \frac{2PR}{P+R}$: combines information on precision and recall

# Evaluating sequence segmentation

WestlakeNLP

- Example:

  Input: 南京市里面和米很贵

  Gold output $S_g$ : '南京市' , '里' , '面' , '和' , '米' , '很' , '贵' (Length: 7)

  System output S: '南京市' , '里面' , '和' , '米' , '很' , '贵' (Length: 6)

  Common subset of segments S : '南京市' , '和' , '米' , '很' , '贵' (Length: 5)

Precision: $P = \frac{S_m}{S} = \frac{5}{6} = 0.83$

Recall: $R = \frac{S_m}{S_g} == \frac{5}{7} = 0.71$

F-score: $F = \frac{2PR}{P+R} = 0.77$

# Contents

WestlakeNLP

# Segmentation vs Sequence Labelling

WestlakeNLP

- Connections
  - Sequence Labelling can be applied to solve sequence segmentation task
  - Output form
    - segment sequence vs. label sequence
    - Transform segmentation into labels.
      - e.g., Segment(S) / attach(A)

        ###　　## 　#

        SAA　SA　S

# Segmentation vs Sequence Labelling

WestlakeNLP

**Sequence Labeling**

- Input: [我] [吃] [了] [苹果]
- Output: 我(P) 吃(V) 了(U) 苹果(NN)
- Labels: P V U NN

**Sequence Segmentation**

- Input: [我] [吃] [了] [苹果]
- Output: [我] [吃 了 苹果]
- Labels: S B-VP I-VP I-VP

- More fine grained tags.
- Combine segmentation label with chunk type.

# Typical label sets

- Word segmentation
  - label: B (Beginning), I (Internal), E (Ending) and S (Single-character word)
- Syntactic chunking
  - label: {B, I}
  - combine syntactic categories: such as B-VP or I-NP
- Named entity recognition
  - label: {B-X, I,E, S-X,O}
  - X indicates the type of entity: PER (person), LOC (location), ORG (organization)
  - O: a non-named entity word

# Features templates

WestlakeNLP

- For discriminative models
  - $score(T_{1:n}, X_{1:n}) = \vec{\theta} \cdot \vec{\phi}(T_{1:n}, X_{1:n})$
- $\vec{\phi}(T_{1:n}, X_{1:n}) = \sum_{i=1} \vec{\phi}(t_i, T_{i-k:i-1}, X_{1:n})$
- Feature templates --- patterns. (e.g., $w_i t_i$)
- Feature instances
  - matching templates to data.
- Feature vector. “He visited New Zealand.”

B-LOC E-LOC

$$< 0, 0, \ldots, 0, 1, 0, \ldots, 0, 1, 0, \ldots, 0, \ldots, 0, \mathrm{I}, 0, \ldots, 0 >$$

$w =$ New
$t =$ E-LOC

$w =$ New
$t =$ B-LOC

$w =$ old
$t =$ B-PER

$w =$ Zealand
$t =$ E-LOC

# Features for word segmentation

WestlakeNLP

| ID | Feature templates | ID | Feature templates |
|---|---|---|---|
| 1 | $c_{i-1}$, $c_i$, $c_{i+1}$ | 4 | $c_{i-1}c_ic_{i+1}$ |
| 2 | $c_{i-1}c_i$, $c_ic_{i+1}$ | 5 | $\textsc{Punc}(c_i)$ |
| 3 | $c_{i-1}c_{i+1}$ | 6 | $\textsc{Type}(c_{i-1})\textsc{Type}(c_i)\textsc{Type}(c_{i+1})$ |

All combine with $t_i$

- $c_i$ represents the $i$-th character in the input sequence
- PUNC indicates whether a character is a punctuation or not
- TYPE indicates the category of a character among four predefined character classes
    - numbers, date time indicators (“年” (year), “月” (month), “日” (day) “时” (hour) “分” (minute) and “秒” (second)), English letters and other characters.

# Example

Input: 其中外企6个

$c_i = c_4 =$‘企’, $t_4 =$‘B'

| ID | Feature Templates | Feature Instances |
| --- | --- | --- |
| 1 | $c_{i-1}, c_i, c_{i+1}$ | '外', '企', '6' |
| 2 | $c_{i-1}c_i, c_ic_{i+1}$ | '外企', '企6' |
| 3 | $c_{i-1}c_{i+1}$ | '外6' |
| 4 | $c_{i-1}c_ic_{i+1}$ | '外企6' |
| 5 | PUNC($c_i$) | False |
| 6 | TYPE($c_{i-1}$)TYPE($c_i$)TYPE($c_{i+1}$) | 'OTHER' 'OTHER' 'NUMBER' |

All combine with “B”

# Features for syntactic chunking

| ID | Feature templates | ID | Feature templates |
|---|---|---|---|
| 1 | $w_{i-2}, w_{i-1}, w_i, w_{i+1}, w_{i+2}$ | 4 | $p_{i-1}p_i, p_ip_{i+1}, p_{i-1}p_{i+1}$ |
| 2 | $p_{i-2}, p_{i-1}, p_i, p_{i+1}, p_{i+2}$ | 5 | $w_{i-1}p_{i-1}, w_ip_i, w_{i+1}p_{i+1}$ |
| 3 | $w_{i-1}w_i, w_iw_{i+1}, w_{i-1}w_{i+1}$ | 6 | $t_{i-1}t_i$ |

- Template 1-5 all combine with $t_i$
- $w_i$ indicates the $i$-th input word
- $p_i$ indicates the POS tag of the $i$-th word
- $t_i$indicates the $i$-th output segmentation label
- Output tag-tag transition features $t_{i-1}$ $t_i$ are useful for syntactic chunking

  e.g. previous chunking label is I-VP, the probability of the next label being I-VP or B-NP can be relatively higher.

# Features for syntactic chunking

WestlakeNLP

Input: Mary went to Chicago to **meet** her boyfriend John Smith.

$w_i = w_6 =$'**meet**'. $t_6 =$'**B-VP**'

| ID | Feature Templates | Feature Instances |
|---|---|---|
| 1 | $w_{i-2}, w_{i-1}, w_i, w_{i+1}, w_{i+2}$ | 'Chicago', 'to', 'meet', 'her', 'boyfriend' |
| 2 | $p_{i-2}, p_{i-1}, p_i, p_{i+1}, p_{i+2}$ | 'NNP', 'TO', 'VB', 'PRP$', 'NN' |
| 3 | $w_{i-1}w_i, w_iw_{i+1}, w_{i-1}w_{i+1}$ | 'to meet', 'meet her', 'to her' |
| 4 | $p_{i-1}p_i, p_ip_{i+1}, p_{i-1}p_{i+1}$ | 'TO VB', 'VB PRP$', 'TO PRP$' |
| 5 | $w_{i-1}p_{i-1}, w_ip_i, w_{i+1}p_{i+1}$ | 'to TO', 'meet VB', 'her PRP$' |
| 6 | $t_{i-1}t_i$ | 'B-PP B-VP' |

All combine with "B-VP"

# Features for NER

WestlakeNLP

| textbfID | **Feature templates** |
|---|---|
| 1 | $w_{i-2}$, $w_{i-1}$, $w_i$, $w_{i+1}$, $w_{i+2}$ |
| 2 | $\textsc{PoS}(w_{i-2})$, $\textsc{PoS}(w_{i-1})$, $\textsc{PoS}(w_i)$, $\textsc{PoS}(w_{i+1})$, $\textsc{PoS}(w_{i+2})$ |
| 3 | $\textsc{Prefix}(w_i)$, $\textsc{Suffix}(w_i)$ |
| 4 | $\textsc{Case}(w_i)$ |
| 5 | $\textsc{Hyphen}(w_i)$ |
| 6 | $\textsc{Shape}(w_{i-2})$, $\textsc{Shape}(w_{i-1})$, $\textsc{Shape}(w_i)$, $\textsc{Shape}(w_{i+1})$, $\textsc{Shape}(w_{i+2})$ |
| 7 | $\textsc{ShortShape}(w_{i-2})$, $\textsc{ShortShape}(w_{i-1})$, $\textsc{ShortShape}(w_i)$, $\textsc{ShortShape}(w_{i+1})$, $\textsc{ShortShape}(w_{i+1})$ |
| 8 | $\textsc{Gazetteer}(w_i)$ |

All combine with "$t_i$"

- Word shape
  - Simplify the word form to reduce sparsity
  - X/x: upper/lower case letters, d: numerical digits
  - Shape($w_i$ = "ELMo") = "XXXx", shortshape( $w_i$ ="ELMo")=Xx.
- Gazetteer features
  - whether the current word exists in a list of known person names, geolocation names, organization names etc.
  - useful for restricted domains

# Features for NER

WestlakeNLP

Input: Mary went to **Chicago** to meet her boyfriend John Smith.

$w_i = w_4 =$'Chicago', $t_4 =$'B-LOC'

| ID | Feature Templates | Feature Instances |
|---|---|---|
| 1 | $w_{i-2}, w_{i-1}, w_i, w_{i+1}, w_{i+2}$ | 'went', 'to', 'Chicago', 'to', 'meet' |
| 2 | POS($w_{i-2}$), POS($w_{i-1}$), POS($w_i$), POS($w_{i+1}$), POS($w_{i+2}$) | 'VBD', 'TO', 'NNP', 'TO', 'VB' |
| 3 | PREFIX($w_i$), SUFFIX($w_i$) | "C"/"Ch“, "g"/"go" |
| 4 | ALL_LOWER_CASE($w_i$) | False |
| 5 | CONTAINS_HYPHEN($w_i$) | False |
| 6 | SHAPE($w_{i-2}$), SHAPE($w_{i-1}$), SHAPE($w_i$), SHAPE($w_{i+1}$), SHAPE($w_{i+2}$) | 'xxxx', 'xx', 'Xxxxxxx', 'xx', 'xxxx' |
| 7 | SHORTSHAPE($w_{i-2}$), SHORTSHAPE($w_{i-1}$), SHORTSHAPE($w_i$), SHORTSHAPE($w_{i+1}$), SHORTSHAPE($w_{i+2}$) | 'x', 'x', 'Xx', 'x', 'x' |
| 8 | GAZETTEER($w_i$) | True |

All combine with “B-LOC”

# Contents

WestlakeNLP

# Problem of Segmentation by Sequence Labelling

WestlakeNLP

Feature vector is the key to discriminative models.

For efficient decoding and training, sequence labelling models assume

**Markov properties over output label sequences**

- A second-order Markov model allows features to be defined over three consecutive segmentation labels
- But segment level features can be beyond label n-grams. There can be words with than three characters. For example, “the previous word = 萧规曹随(to follow convention)” cannot be directly modeled.

# Directly Modeling for Segmentation

WestlakeNLP

Model sequence segmentation directly using discriminative structured predictors, which score output sequences  with segment-level features

- As extensions to discriminative sequence labelers for a different output structure – **sequence segmentation**
- We consider discriminative models in this chapter.
- Three aspects to discuss in detail
  - segment-level feature definitions
  - decoding
  - training

# Contents

WestlakeNLP

# Word-Level Features for Word Segmentation

WestlakeNLP

- Take Chinese word segmentation as an example task
- Suppose that features are defined within two consecutive words, or a word bigram
- For an input sentence $C_{1:n} = c_1 c_2 \ldots c_n$, a segmented output can be denoted as $W_{1:|W|} = w_1 w_2 \ldots w_{|W|}$
  - $w_j = c_{b(j)} c_{b(j)+1} \ldots c_{e(j)}$
  - $b(j)$ and $e(j)$ denote the character indices for the first and last characters in the word $w_j$

# Word-Level Features for Word Segmentation

WestlakeNLP

- E.g., 我 昨天 打球 了 $w_2 =$ 昨天, $b(2) = 2, e(2) = 3$

- Global feature vector $\vec{\phi}(W_{1:|W|})$ can be extracted by accumulating local features $\vec{\phi}(w_{j-1}, w_j)$ over all word bigrams $w_{j-1}w_j$ in the output sequence:

$$\vec{\phi}(W_{1:|W|}) = \sum_{j=2}^{|W|} \vec{\phi}(w_{j-1}, w_j)$$

- $\vec{\phi}(w_{j-1}, w_j) \equiv \overrightarrow{\phi_c}(C_{1:n}, b(j-1), e(j-1), e(j))$

# Word-Level Features for Word Segmentation

WestlakeNLP

| ID | Feature templates | ID | Feature templates |
|---|---|---|---|
| 1 | word $w_j$ | 8 | $c_{b(j)}c_{e(j)}$ |
| 2 | word bigram $w_{j-1}w_j$ | 9 | $w_j c_{e(j)+1}$ |
| 3 | whether $w_j$ is a single-character word, $\text{SINGLE}(w_j)$ | 10 | $w_j c_{e(j-1)}$ |
| 4 | $c_{b(j)}\text{LEN}(w_j)$ | 11 | $c_{b(j-1)}c_{b(j)}$ |
| 5 | $c_{e(j)}\text{LEN}(w_j)$ | 12 | $c_{e(j-1)}c_{e(j)}$ |
| 6 | space-separated characters, $c_{e(j-1)}c_{b(j)}$ | 13 | $w_j\text{LEN}(w_{j-1})$ |
| 7 | character bigram in $w_j$ | 14 | $w_{j-1}\text{LEN}(w_j)$ |

# Example

- Input: <s> 我 吃 了 苹果 </s>

| Feature Entry $\vec{\phi}\,(w_{i-1}, w_i)$ | Feature Vector |
| --- | --- |
| $\vec{\phi}\,(w_0, w_1)$ | 0, 0, …, $f_{30}(w_{i-1}w_i = \text{“}<s>\text{我”}) = 1$, $f_{201}(w_i \textit{ is a single character}) = 1$, … |
| $\vec{\phi}\,(w_1, w_2)$ | 0, 0, …, $f_{47}(w_{i-1}w_i = \text{“我吃”}) = 1$, …, $f_{201}(w_i \textit{ is a single character}) = 1$, … |
| $\vec{\phi}\,(w_2, w_3)$ | 0, 0, …, $f_{51}(w_{i-1}w_i = \text{“吃了”}) = 1$, …, $f_{201}(w_i \textit{ is a single character}) = 1$, … |
| $\vec{\phi}\,(w_3, w_4)$ | 0, 0, …, $f_{472}(w_{i-1}w_i = \text{“了苹果”}) = 1$, … |
| $\vec{\phi}\,(w_4, w_5)$ | 0, 0, …, $f_{501}(w_{i-1}w_i = \text{“苹果}</s>\text{”}) = 1$, … |
| $\vec{\phi}\,(W_{1:4})$ | 0, 0, …, $f_{30}(w_{i-1}w_i = \text{“}<s>\text{我”}) = 1$, …, $f_{47}(w_{i-1}w_i = \text{“我吃”}) = 1$, …, $f_{51}(w_{i-1}w_i = \text{“吃了”}) = 1$, …, $f_{201}(w_i \textit{ is a single character}) = 3$, …, $f_{472}(w_{i-1}w_i = \text{“了 苹果”}) = 1$, …, $f_{501}(w_{i-1}w_i \textit{ is } \text{“苹果}</s>\text{”}) = 1$, … |

# Contents

WestlakeNLP

# Discriminative linear models for sequence segmentation

- Use word segmentation for example
- A discriminative linear model to score different segmentation outputs

  $C_{1:n}$ given an input $W_{1:|w|}$, according to the feature representation $\vec{\phi}(W_{1:|w|})$

  - $Score(W_{1:|w|}) = \vec{\theta} \cdot \vec{\phi}(W_{1:|w|})$
- Two discriminative linear model instances
  - log-linear models (semi-CRF)
  - large margin models (SVM, perceptron)
  - Decoding uses the same algorithms

# Decoding

- $C_{1:n}$: an input sentence
- $W_{1:|w|}$ : an output segmentation
- The goal of decoding is to find the highest-scored output $\widehat{W}$ according to a given model $\vec{\theta}$:

$$\widehat{W} = argmax_w \vec{\theta} \cdot \vec{\phi}(W)$$

- Assume that features are extracted from word bigrams

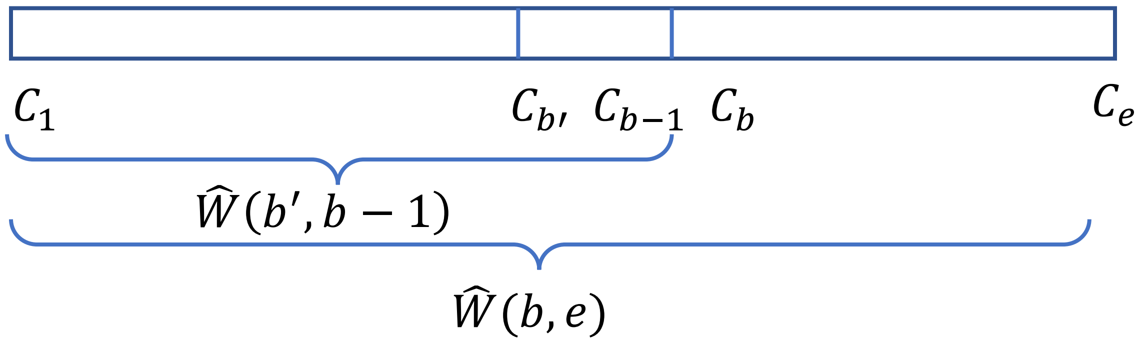
$$\vec{\theta} \cdot \vec{\phi}(W_{1:|w|}) = \vec{\theta} \cdot \left( \sum_{j=2}^{|W|} \vec{\phi}(w_{j-1}, w_j) \right) = \sum_{j=2}^{|W|} \vec{\theta} \cdot \vec{\phi}(w_{j-1}, w_j) = \sum_{j=2}^{|W|} \vec{\theta} \cdot \overrightarrow{\phi_c}(C_{1:n}, b(j-1), e(j-1), e(j))$$

- Score can be computed incrementally adding word by word

# Decoding

WestlakeNLP

- Denote a word sequence with the last word being $C_{b:e}$ as *W(b, e)*.
- the highest scored output sequence with the last word being $C_{b:e}$ as $\widehat{W}(b, e)$.
- Suppose that the second last word in $\widehat{W}(b, e)$ is $C_{b':b-1}$
- Then the subsequence in $\widehat{W}(b, e)$ that ends with $c_{b-1}$ must be the highest-scored among all segmentation sequences that end with $C_{b':b-1}$, namely $\widehat{W}(b', b-1)$.




- Therefore a table can be built for $\widehat{W}(b, e)$ incrementally.

# Decoding

The incremental nature of the score calculation results in the availability of optimal sub problems (DP):

$$score\left(\widehat{W}(b,e)\right)$$
$$= argmax_{1\le b'\le b-1}\left(score(\widehat{W}(b',b-1)) + \vec{\theta}\cdot\overrightarrow{\phi_c}(C_{1:n},b',b-1,e)\right)$$

- $\widehat{W}(b,e)$ denotes the highest-scored partial output with the last word being $C_{b:e} = c_b, c_{b+1} \dots c_e$
- the beginning character index $b \in [1 \dots n]$
- the ending character index $e \in [b \dots n]$.

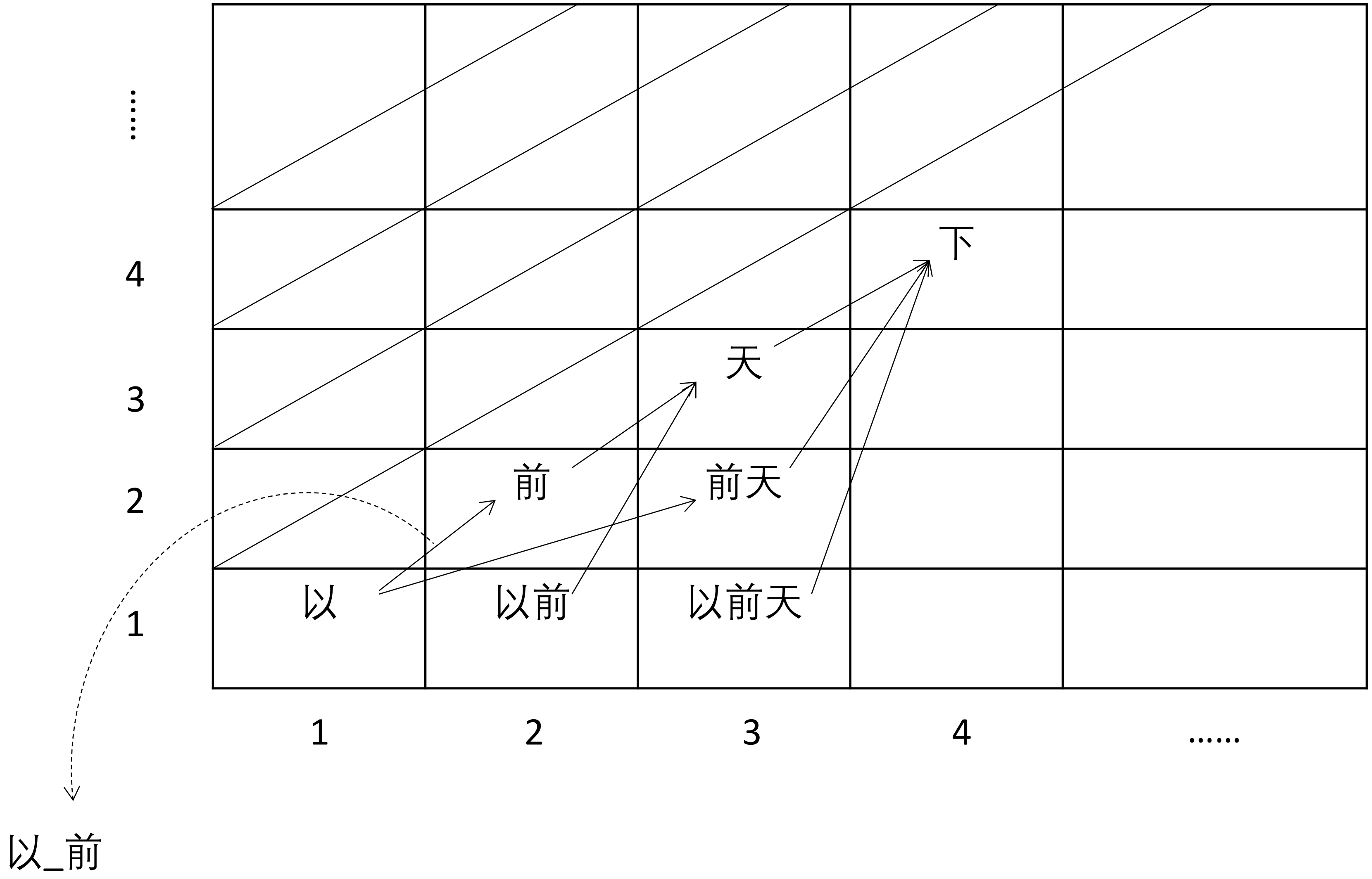
# Decoding

$$score\left(\widehat{W}(b,e)\right) = argmax_{1\le b'\le b-1}\left(score(\widehat{W}(b',b-1)) + \vec{\theta}\cdot\overrightarrow{\phi_c}(C_{1:n},b',b-1,e)\right)$$

- Use table to store $score(\widehat{W}(b,e))$ for all $b \in [1,\dots,n], e \in [b,\dots,n]$
- Use bp to store $argmax_{b'}$.
- Both $n\times n$ in size.
- The final highest-scored output:

$$\widehat{W} = argmax_{b\in[1\dots n]} score\left(\widehat{W}(b,n)\right)$$

# Decoding



# Decoding

WestlakeNLP

**Input**: Sequence $C_{1:n} = c_1 c_2 \ldots c_n$, model parameters $\vec{\theta}$;
**Initialisation:**
**for** $e \in [1, \ldots, n]$ **do**
  **for** $b \in [1, \ldots, e]$ **do**
    $table[b, e] \leftarrow -\infty$;
    $bp[b, e] \leftarrow -1$;
  $table[1, e] \leftarrow \vec{\theta} \cdot \vec{\phi}_c(C_{1:n}, 0, 0, e)$;
**Algorithm:**
**for** $e \in [2, \ldots, n]$ **do**
  **for** $b \in [2, \ldots, e]$ **do**
    **for** $b' \in [1, \ldots, b-1]$ **do**
      **if** $table[b', b-1] + \vec{\theta} \cdot \vec{\phi}_c(C_{1:n}, b', b-1, e) > table[b, e]$ **then**
        $table[b, e] \leftarrow table[b', b-1] + \vec{\theta} \cdot \vec{\phi}_c(C_{1:n}, b', b-1, e)$;
        $bp[b, e] \leftarrow b'$;
$max\_score \leftarrow \max_{b'} table[b', n] + \vec{\phi}_c(C_{1:n}, b', n, n+1)$;
backtrace with $bp$;
**Output**: Segmented sequence $W_{1:|W|} = w_1 w_2 \ldots w_{|W|}$;

- The complexity is O($n^3$), due to the enumeration of e, b and b'
- Force a maximum word size M: linear time complexity

# Contents

# Semi-Markov Conditional Random Fields

WestlakeNLP

- Semi-CRF is a log-linear model for sequence segmentation, which gives a probability interpretation to the scores assigned to segmented output structures.

$$P(W|C) = \frac{\exp\left(\vec{\theta} \cdot \vec{\phi}(W)\right)}{\sum_{W' \in \mathrm{GEN}(C)} \exp\left(\vec{\theta} \cdot \vec{\phi}(W')\right)}$$

  $\mathrm{GEN}(C)$ denotes all possible segmented outputs of $C$

- We discuss below:
  - Calculating marginal probabilities
  - Training a CRF model

# Calculating Marginal Probabilities

WestlakeNLP

- Given an input $C_{1:n}$, denote the probability of $C_{b:e} = c_b c_{b+1} \dots c_e$ being a word as $P(WRD(C_{b:e})|C_{1:n})$, where $WRD(C_{b:e})$ indicates that $C_{b:e}$ is a word in the sentence.
- We want to estimate $P(WRD(C_{b:e})|C_{1:n})$

$$P(WRD(C_{b:e})|C_{1:n}) = \sum_{W \in GEN(C_{1:n}), C_{b:e} \in W} P(W|C_{1:n})$$

- $W \in GEN(C_{1:n}), C_{b:e} \in W$ denotes all possible segmentations of $C_{1:n}$ that contain the word $C_{b:e}$
- An exponential number of summations

# Calculating Marginal Probabilities

WestlakeNLP

Since features are local to word bigrams, we have

$$P(W|C_{1:n}) = \frac{\exp\left(\vec{\theta} \cdot \vec{\phi}(W)\right)}{Z}$$

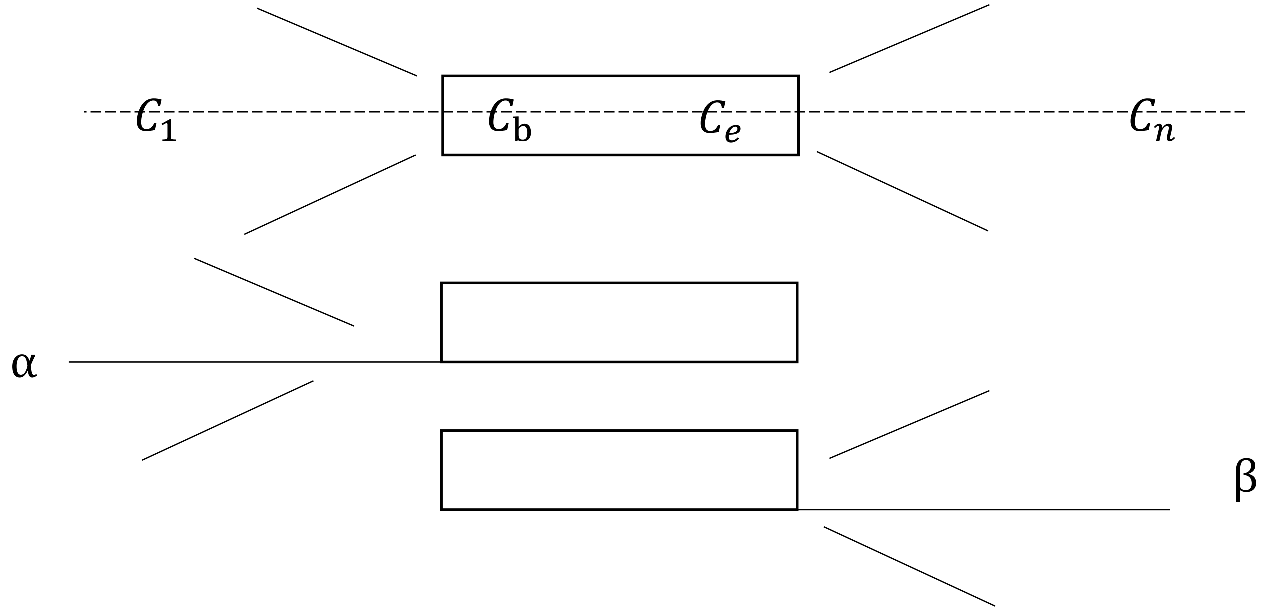
$$= \frac{\exp\left(\vec{\theta} \cdot \left(\sum_j \vec{\phi}(w_{j-1}, w_j)\right)\right)}{Z}$$

$$= \frac{\prod_j \exp\left(\vec{\theta} \cdot \vec{\phi}(w_{j-1}, w_j)\right)}{Z}$$

where $Z$ is the partition function $\sum_W \exp\left(\vec{\theta} \cdot \vec{\phi}(W)\right)$.

# Calculating Marginal Probabilities

$$P(WRD(C_{b:e})|C_{1:n}) = \sum_{W \in GEN(C_{1:n}), c_{b:e} \in W} \left( \frac{1}{Z} \prod_{j=1:|W|} \exp\left(\vec{\theta} \cdot \vec{\phi}(w_{j-1}, w_j)\right) \right)$$



# Calculating Marginal Probabilities

WestlakeNLP

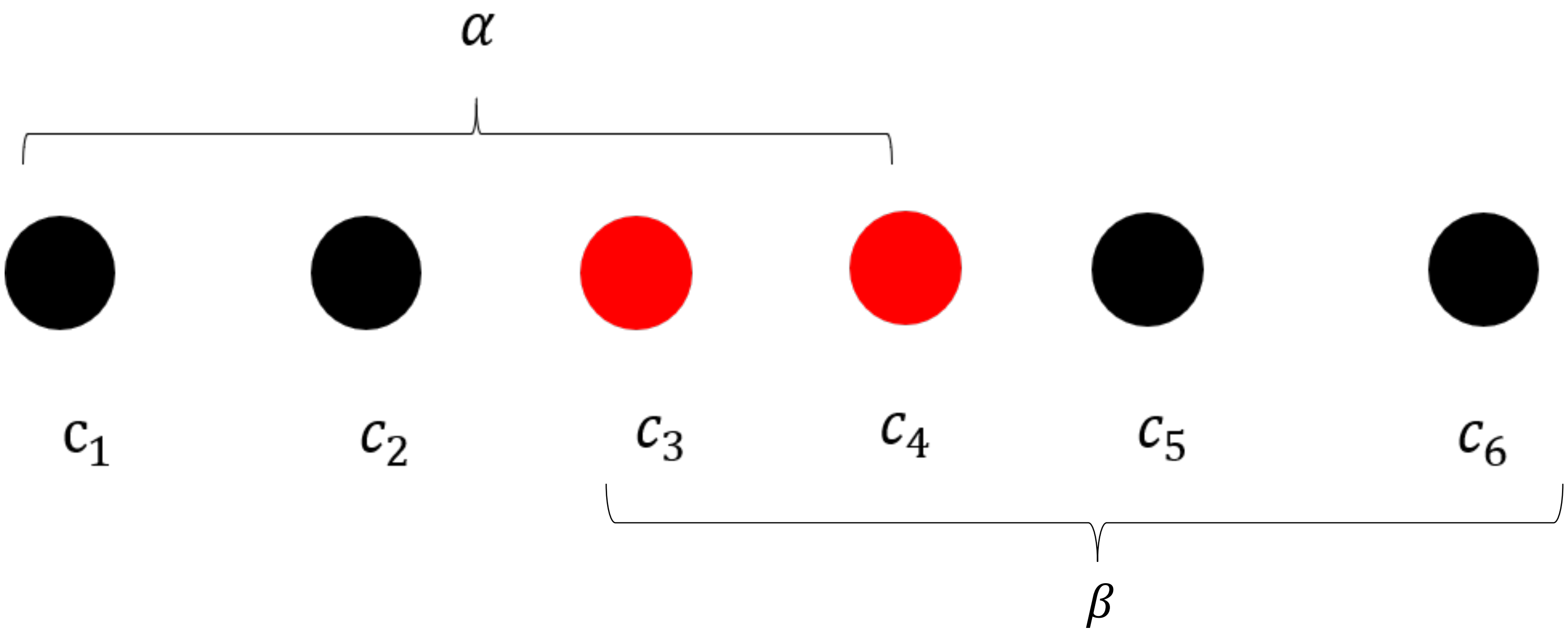
$$P(\textsc{IsWord}(C_{b:e})|C_{1:n}) = \sum_{W \in \textsc{Gen}(C_{1:n}) \text{ such that } C_{b:e} \in W} \left( \frac{1}{Z} \prod_{j=1}^{|W|} \exp\left(\vec{\theta} \cdot \vec{\phi}(w_{j-1}, w_j)\right) \right)$$

$$= \frac{1}{Z} \left( \sum_{W^l \in \textsc{Gen}(C_{1:e}) \text{ such that } C_{b:e} \in W^l} \prod_{j=1}^{|W^l|} \exp\left(\vec{\theta} \cdot \vec{\phi}(w^l_{j-1}, w^l_j)\right) \right) \Rightarrow \alpha(b,e)$$

$$\left( \sum_{W^r \in \textsc{Gen}(C_{b:n}) \text{ such that } C_{b:e} \in W^r} \prod_{j=1}^{|W^r|-1} \exp\left(\vec{\theta} \cdot \vec{\phi}(w^r_j, w^r_{j+1})\right) \right) \Rightarrow \beta(b,e)$$

- For $W^l = w^l_1, w^l_2, \dots, w^l_{|W^l|}$, $w^l_{|W^l|} = C_{b:e}$
- For $W^r = w^r_1, w^r_2, \dots, w^r_{|W^r|}$, $w^r_1 = C_{b:e}$
- cuts the full summation into the product of two components, with the splitting point at $(b, e)$.

# Calculating Marginal Probabilities




- $C_{b:e} = C_{3:4}$
- It's similar to Forward-Backward Algorithm in CRF

# Forward Algorithm for semi-CRF

WestlakeNLP

- For the first component

$$\alpha(b',e') = \sum_{W^l \in GEN(C_{1:e'}), C_{b':e'} \in W^l} \prod_{j=1}^{|W^l|} \exp\left(\vec{\theta} \cdot \vec{\phi}(w_{j-1}^l, w_j^l)\right) =$$

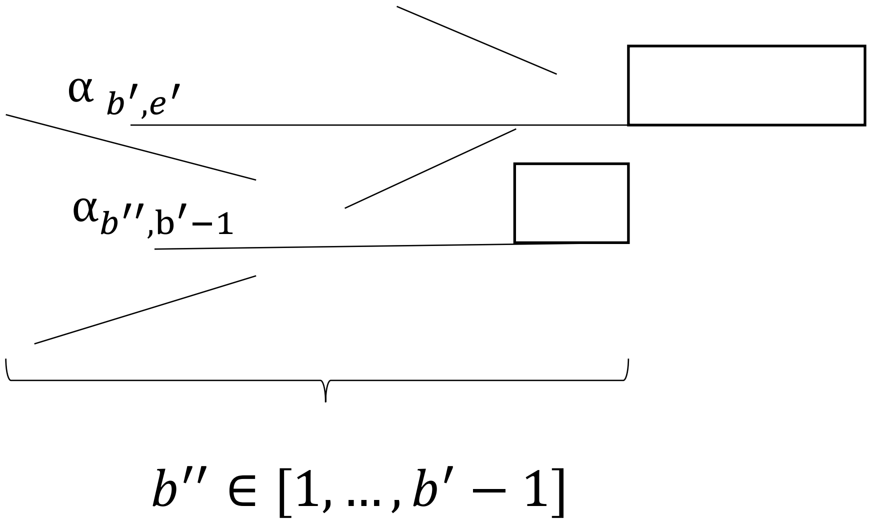
$$\sum_{b'' \in [1 \ldots b'-1]} \sum_{w^e \in GEN(C_{1:e'}), C_{b'':b'-1} \in w^e} \prod_{j=1}^{|w^e|} \exp\left(\hat{\theta} \cdot \hat{\phi}(w_{j-1}^l, w_j^l = C_{b'',b'-1})\right) \cdot \exp\left(\hat{\theta} \cdot \hat{\phi}(C_{b'',b'-1}, C_{b',e})\right)$$

- $\alpha(b',e')$ can be calculated incrementally by summing up relevant values regarding $\alpha(b'',b'-1)$ for all valid $b''$ :

$$\alpha(b',e') = \sum_{b'' \in [1 \ldots b'-1]} \left(\alpha(b'',b'-1) \cdot \exp\left(\vec{\theta} \cdot \overrightarrow{\phi_c}(C_{1:e}, b'', b'-1, e')\right)\right)$$

where $b' \in [1, \ldots, e], e' \in [b', \ldots, e]$

# Forward Algorithm for semi-CRF




$$\alpha(b', e') = \sum_{b'' \in [1 \dots b'-1]} \left( \alpha(b'', b'-1) \cdot \exp\left( \vec{\theta} \cdot \overrightarrow{\phi_c}(C_{1:e}, b'', b'-1, e') \right) \right)$$

where $b' \in [1, \dots, e], e' \in [b', \dots, e]$

# Forward Algorithm for semi-CRF

WestlakeNLP

**Inputs**: $s = C_{1:e}$, semi-CRF model with feature weight vector $\vec{\theta}$;
**Variables**: $\alpha$;
**Initialisation:**
**for** $e' \in [1, \ldots, e]$ **do**
  $\alpha[1, e'] \leftarrow \vec{\theta} \cdot \vec{\phi}(C_{1:e}, 0, 0, e')$;
**Algorithm:**
**for** $b \in [2, \ldots, e]$ **do**
  **for** $e \in [b', \ldots, e]$ **do**
    $\alpha[b', e'] \leftarrow 0$;
    **for** $b'' \in [1, \ldots, b' - 1]$ **do**
      $\alpha[b', e'] \leftarrow$
      $\alpha[b', e'] + \alpha[b'', b' - 1] \cdot \exp\left(\vec{\theta} \cdot \vec{\phi}_c(C_{1:n}, b'', b' - 1, e')\right)$;
**Output**: $\alpha$;

- Starting from boundary values

$$\alpha(1, e') = \exp\left(\vec{\theta} \cdot \overrightarrow{\phi_c}(C_{1:e}, 0,0, e')\right) \text{ for } e' \in [1, \ldots, e],$$

# Backward Algorithm for semi-CRF

WestlakeNLP

- For the second component

$$\beta(b', e') = \sum_{W^r \in GEN(C_{b':n}), C_{b':e'} \in W^r} \prod_{j=1}^{|W^r|-1} \exp\left(\vec{\theta} \cdot \vec{\phi}(w_j^r, w_{j+1}^r)\right)$$

- $\beta(b', e')$ can be calculated incrementally by summing up relevant values from all $\beta(e'+1, e'')$, where $e'' \in [e'+1, \dots, n]$

$$\beta(b', e') = \sum_{e'' \in [e'+1, \dots, n]} \left(\beta(e'+1, e'') \cdot \exp\left(\vec{\theta} \cdot \overrightarrow{\phi_c}(C_{e+1:n}, b', e', e'')\right)\right)$$

where $b' \in [e+1, \dots, n], e' \in [e+1, \dots, n]$.

# Backward Algorithm for semi-CRF

**Inputs**: $s = C_{b:n}$, semi-CRF model with feature weight vector $\vec{\theta}$;
**Variables**: $\beta$;
**Initialisation:**
**for** $b' \in [n, n-1, \ldots, b]$ **do**
  $\beta[b', n] \leftarrow 1$;
**Algorithm:**
**for** $e' \in [n-1, n-2, \ldots, b]$ **do**
  **for** $b' \in [e', e'-1, \ldots, b]$ **do**
    $\beta[b', e'] \leftarrow 0$;
    **for** $e'' \in [e'+1, \ldots, n]$ **do**
      $\beta[b', e'] \leftarrow \beta[b', e'] + \beta[e'+1, e''] \cdot \exp\left(\vec{\theta} \cdot \vec{\phi}_c(C_{b:n}, b', e', e'')\right)$;
**Output**: $\beta$;

- Starting from boundary values

$$\beta(b', n) = 1$$

# Calculating Marginal Probabilities

WestlakeNLP

- After obtaining $\alpha(b', e')$ and $\beta(b', e')$ values, $P\left(WRD(C_{b:e}|C_{1:n})\right)$ can be calculated as:

$$\frac{1}{Z}\alpha(b,e)\beta(b,e)$$

# Partition function for semi-CRF

- Partition Function

$$Z = \sum_{w} \exp(\hat{\theta} \cdot \hat{\phi}(w))$$

- Can use a dynamic program, similar to the decoding algorithm, but with *max* being replaced by *sum*.

# Partition function for semi-CRF

WestlakeNLP

**Inputs:** $s = C_{1:n}$, semi-CRF model model and feature weight vector $\vec{\theta}$;
**Initialisation:**
**for** $e \in [1, \ldots, n]$ **do**
    $table[1, e] \leftarrow \vec{\theta} \cdot \vec{\phi}_c(C_{1:n}, 0, 0, e)$;
**Algorithm:**
**for** $e \in [2, \ldots, n]$ **do**
    **for** $b \in [2, \ldots, e]$ **do**
        $scores \leftarrow []$;
        **for** $b' \in [1, \ldots, b-1]$ **do**
            $\textsc{Append}\Big(scores,\ table[b', b-1] + \vec{\theta} \cdot \vec{\phi}_c(C_{1:n}, b', b-1, e)\Big)$;
        $table[b, e] \leftarrow logsumexp(scores)$;
$Z \leftarrow \sum_{b \in [1,\ldots,n]} \exp(table[b, n])$;
**Output:** $Z$;

- Log sum exp trick can be used to avoid numeric overflow.

# Contents

WestlakeNLP

# Training semi-CRF

WestlakeNLP

Given a set of training data $D = \{(C_i, W_i)\}|_{i=1}^{n}$, where $C_i$ is a sentence and $W_i$ is its corresponding gold-standard segmentation, the semi-CRF training objective is to maximize the log-likelihood of $D$:

$$\hat{\vec{\theta}} = argmax_{\vec{\theta}} \log P\,(D)$$

$$= argmax_{\vec{\theta}} \sum_i \log P\,(W_i|C_i)$$

$$= argmax_{\vec{\theta}} \sum_i \log \frac{\exp\left(\vec{\theta} \cdot \vec{\phi}(W_i, C_i)\right)}{\sum_{W' \in GEN(W)} \exp\left(\vec{\theta} \cdot \vec{\phi}(W', C_i)\right)}$$

$$= argmax_{\vec{\theta}} \sum_i \left(\vec{\theta} \cdot \vec{\phi}(W_i, C_i) - \log\left(\sum_{W' \in GEN(W_i)} \exp\left(\vec{\theta} \cdot \vec{\phi}(W', C_i)\right)\right)\right)$$

# Local gradient

# Local gradient

- For each training example, the local gradient with respect to $\vec{\theta}$ is:

$$\vec{\phi}(W_i, C_i) - \frac{\sum_{W'} \exp\left(\vec{\theta}\cdot\vec{\phi}(W',C_i)\right)\cdot\vec{\phi}(W',C_i)}{\sum_{W''} \exp\left(\vec{\theta}\cdot\vec{\phi}(W'',C_i)\right)}$$

$$= \vec{\phi}(W_i, C_i) - \sum_{W'} P(W'|C_i)\vec{\phi}(W', C_i)\,,\left(\text{definition of } P(W'|C_i)\right)$$

- The major challenge is the summation of exponential possible outputs.

# Local gradient

WestlakeNLP

- Similar to CRF, rely on feature locality.

  Taking word segmentation for example:

$$\sum_{W'} P(W'|C_i)\vec{\phi}(W', C_i) = \sum_{W' \in GEN(C_i)} P(W'|C_i)\left(\sum_{j=1}^{|W'|} \vec{\phi}(w_{j-1}, w_j)\right)$$

$$= E_{W' \sim P(W'|C_i)}\left(\sum_{j=1}^{|W'|} \vec{\phi}(w_{j-1}, w_j)\right)$$

# Solution: feature locality

WestlakeNLP

- We can rewrite $\sum_{W'} P(W'|C_i)$ as:
- $E_{W' \sim P(W'|C_i)}\left(\sum_{j=1}^{|W'|} \vec{\phi}(w_{j-1}, w_j)\right)$

$$= E_{W' \sim P(W'|C_i)}\left(\sum_{C_{b':b-1} \in W', C_{b:e} \in W'} \overrightarrow{\phi_c}(C_i, b', b-1, e)\right)$$

$$= \sum_{b',b,e} E_{C_{b':b-1}\, C_{b:e} \sim P(\text{IsBigram}(b',b-1,e)|C_i)} \overrightarrow{\phi_c}(C_i, b', b-1, e)$$

- GENBIGRAM represents the set of all bigrams in all possible segmentations of $C_i$

# Solution: feature locality

- Equal to the sum of the feature vectors weighed by the marginal probability of the bigram: $P(\text{IsBigram}(b', b-1, e)|C_i)\overrightarrow{\phi_c}(C_i, b', b-1, e)$

- Thus, the task boils down to the calculation of the marginal probabilities $P(BIGRAM(b', b-1, e)|C_i)$ efficiently for all valid values of $b'$, $b$ and $e$

# Solution: feature locality

# Solution: feature locality

$$P(\textsc{IsBigram}(b', b-1, e) | C_i)$$

$$= \sum_{W \in \textsc{Gen}(C_i),\ \text{such that } C_{b':b-1} \in W, C_{b:e} \in W} \frac{1}{Z} \prod_{j=1}^{|W|} \exp\left(\vec{\theta} \cdot \vec{\phi}(w_{j-1}, w_j)\right)$$

$$= \frac{1}{Z} \left( \sum_{W^l \in \textsc{Gen}(C_{1:b-1}),\ \text{such that } C_{b':b-1} \in W^l} \prod_{j=1}^{|W^l|} \exp\left(\vec{\theta} \cdot \vec{\phi}(w^l_{j-1}, w^l_j)\right)\right)$$

$$\left( \sum_{W^r \in \textsc{Gen}(C_{b:n}),\ \text{such that } C_{b:e} \in W^r} \prod_{j=1}^{|W^r|-1} \exp\left(\vec{\theta} \cdot \vec{\phi}(w^r_j, w^r_{j+1})\right)\right),$$

- For $W^l$, we have $W^l_{|W^l-1|} = C_{b':b-1}$, and for $W^r$, we have $w^r_1 = C_{b:e}$

# Solution: feature locality

WestlakeNLP

$P(BIGRAM(b', b-1, e)|C_i)$ can be computed efficiently using Forward-Backward technique

$$P(BIGRAM(b', b-1, e)|C_i)$$

$$= \frac{\alpha(b', b-1)\beta(b, e)\exp\left(\vec{\theta} \cdot \overrightarrow{\phi_c}(C_i, b', b-1, e)\right)}{Z}$$

# Forward Backward Algorithm for training semi-CRF

**Inputs:** $s = C_{1:n}$, semi-CRF model with feature weight vector $\vec{\theta}$;
**Variables:** $table$, $\alpha, \beta$;
$\alpha \leftarrow \text{FORWARD}(C_{1:n}, \vec{\phi}, \vec{\theta})$ u
$\beta \leftarrow \text{BACKWARD}(C_{1:n}, \vec{\phi}, \vec{\theta})$ u
$Z \leftarrow \text{PARTITION}(C_{1:n}, \vec{\phi}, \vec{\theta})$ u
**for** $b \in [1, \ldots, n]$ **do**
  **for** $e \in [b, \ldots, n]$ **do**
    **for** $b' \in [1, \ldots, b-1]$ **do**
      $table[b'][b-1][e] \leftarrow$
        $\alpha[b'][b-1] \cdot \beta[b][e] \cdot \exp\left(\vec{\theta} \cdot \vec{\phi}_c(C_{1:n}, b', b-1, e)\right)/Z$;
**Output:** $table$;

# Forward Backward Algorithm for training semi-CRF

WestlakeNLP

- Partition Function

$$Z = \sum_{w} \exp(\hat{\theta} \cdot \hat{\phi}(w))$$

- Can use a dynamic program, similar to the decoding algorithm, but with *max* being replaced by *sum*.

# Partition function for semi-CRF

**Inputs:** $s = C_{1:n}$, semi-CRF model model and feature weight vector $\vec{\theta}$;
**Initialisation:**
**for** $e \in [1, \ldots, n]$ **do**
    $table[1, e] \leftarrow \vec{\theta} \cdot \vec{\phi}_c(C_{1:n}, 0, 0, e)$;
**Algorithm:**
**for** $e \in [2, \ldots, n]$ **do**
    **for** $b \in [2, \ldots, e]$ **do**
        $scores \leftarrow []$;
        **for** $b' \in [1, \ldots, b-1]$ **do**
            $\textsc{Append}\Big(scores,\ table[b', b-1] + \vec{\theta} \cdot \vec{\phi}_c(C_{1:n}, b', b-1, e)\Big)$;
        $table[b, e] \leftarrow logsumexp(scores)$;
$Z \leftarrow \sum_{b \in [1, \ldots, n]} \exp(table[b, n])$;
**Output:** $Z$;

- Log sum exp trick can be used to avoid numeric overflow.

# Contents

WestlakeNLP

# Large Margin Models

- Scoring
  - $score(S) = \vec{\theta} \cdot \vec{\phi}(S)$
- Decoding: same as semi-CRF

# Large Margin Models

WestlakeNLP

- Scoring
  - $score(S) = \vec{\theta} \cdot \vec{\phi}(S)$
- Decoding: same as semi-CRF
- Training
  - largely the same as those for sequence labelling
  - structure perceptron $\sum_{i=1}^{N} \max\left(0, \max_{S'}\left(\vec{\theta} \cdot \vec{\phi}(S')\right) - \vec{\theta} \cdot \vec{\phi}(S_i)\right)$
  - structured SVM

$$\frac{1}{2}\left|\left|\vec{\theta}\right|\right|^2 + C\left(\sum_{i=1}^{N} \max\left(0, 1 - \vec{\theta} \cdot \vec{\phi}(S_i) + \max_{S' \neq S_i}\left(\vec{\theta} \cdot \vec{\phi}(S')\right)\right)\right)$$

# Large Margin Models

**Input:** $D = (x_i, c_i)|_{i=1}^{N}$, $c_i \in C$

**Initialization:** $\vec{\omega} \leftarrow \vec{0}$; $t \leftarrow 0$;

**repeat**

    **for** $i \in [1 \ldots N]$ **do**

        $z_i \leftarrow \arg\max_{\mathbf{z}} \vec{\omega}^T \vec{v}(x_i, \mathbf{z})$ ;

        **if** $z_i \neq c_i$ **then**

            $\vec{\omega} \leftarrow \vec{\omega} + \vec{v}(x_i, c_i) - \vec{v}(x_i, z_i)$;

    $t \leftarrow t + 1$;

**until** $t = T$;

# Contents

# Segment-level features

- Pros
  - offer a wider context range
  - a direct source of information about the output structures
- Cons
  - feature sparsity
    - For syntactic chunking, a possible noun phrase can span over tens of words.
  - decoding inefficiency
    - using segment bigram feature: $O(n^3)$
    - using segment trigram features: $O(n^4)$

# Segment-level features

WestlakeNLP

- Pros
  - offer a wider context range
  - a direct source of information about the output structures
- Cons
  - feature sparsity
    - For syntactic chunking, a possible noun phrase can span over tens of words.
  - **decoding inefficiency**
    - **using segment bigram feature: O($\boldsymbol{n^3}$)**
    - **using segment trigram features: O($\boldsymbol{n^4}$)**

# Solution: beam search

WestlakeNLP

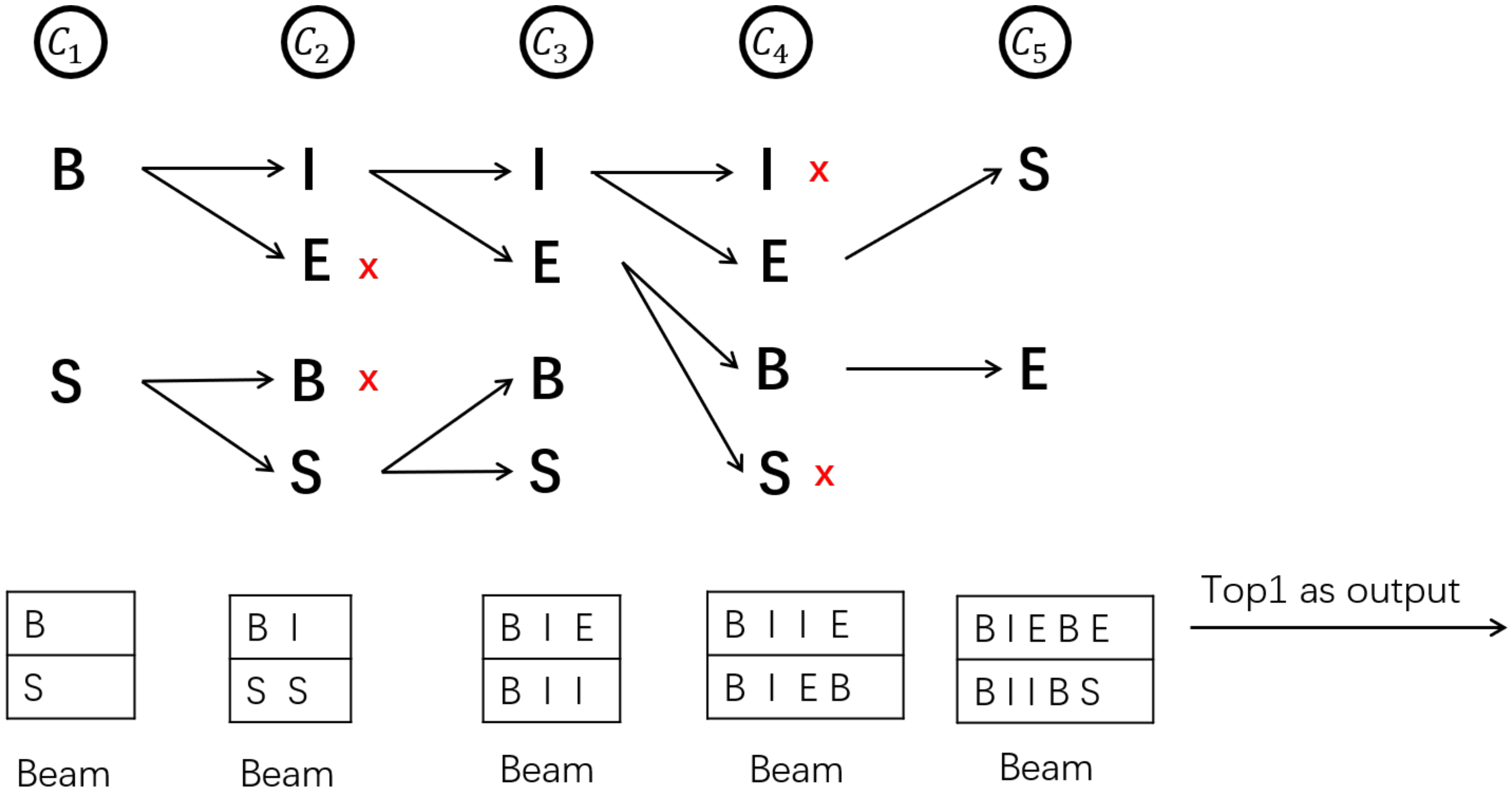
- Model can use arbitrary features without Markov assumptions
- Inexact search to accommodate feature context
- Incrementally processes the input sequence from left to right, building the output structure in linear time.
- Tradeoff between optimality and efficiency.

# Beam Search Decoding

Given an input sentence $W_{1:n}$, the algorithm incrementally builds partial output candidates $T_{1:i}$ from left to right, using an agenda to maintain the $k$ highest scored partial output at each step.

- Each candidate is a partial output $T_{1:i}$.
- Starting from an initial agenda with an empty sequence
- At each step, enumerate all possible local structures concerning the current word to generate new partial output candidates
- Score each candidate and leave top-k candidates for next step
- Repeats until the end of the sentence, the top-1 left is taken for output

# An Example of Beam Search




$c_{1:5}$ = 西 班 牙 足 球

# Beam Search Decoding Algorithm

WestlakeNLP

**Inputs**: $\vec{\theta}$ — discriminative linear model parameters;
$W_{1:n}$ — input sequence;
$k$ — beam size;
**Initialisation**: $agenda \leftarrow [([], 0)]$;
**Algorithm**:
**for** $i \in [1, \ldots, n]$ **do**
  $candidates \leftarrow agenda$;
  $agenda \leftarrow []$;
  **for** $candidate \in candidates$ **do**
    $T_{1:i-1} \leftarrow candidate[0]$;
    $score \leftarrow candidate[1]$;
    **for** $t \in L$ **do**
      $T_1^i \leftarrow \textsc{Expand}(T_{1:i-1}, t)$;
      $new\_score \leftarrow score + \vec{\theta} \cdot \vec{\phi}_\Delta(W_{1:n}, T_{1:i-1}, t)$;
      $\textsc{Append}(agenda, (T_{1:i}, new\_score))$;
  $agenda \leftarrow \textsc{Top-k}(agenda, k)$;
**Output**: $\textsc{Top-k}(agenda, 1)[0]$;

# Relaxing feature locality constraints

WestlakeNLP

At each step, we should score partial outputs from the beginning of the sentence until the current word being processed

- At the $i$-th incremental step, the feature vector for the partial output $T_{1:i}$ is built incrementally from the previous step:

$$\overrightarrow{\phi}(W_{1:n}, T_{1:i}) = \overrightarrow{\phi}(W_{1:n}, T_{1:i-1}) + \overrightarrow{\phi_\Delta}(W_{1:n}, T_{1:i-1}, t_i)$$

- $\overrightarrow{\phi_\Delta}(W_{1:n}, T_{1:i-1}, t_i)$ indicates the incremental feature vector that consists of the partial structures concerning $t_i$
- Differences from the incremental feature for sequence labeling $\overrightarrow{\phi}(W_{1:n}, T_{I-k:i-1}, t_i)$
  - no Markov restriction on the label context

# Contents

WestlakeNLP

# Beam Search

- Problems
  - highest model score is not guaranteed to be found by the decoder.
- Solution
  - adjust the training objective into the minimization of search errors.
  - Merge model error and search error into one single objective.

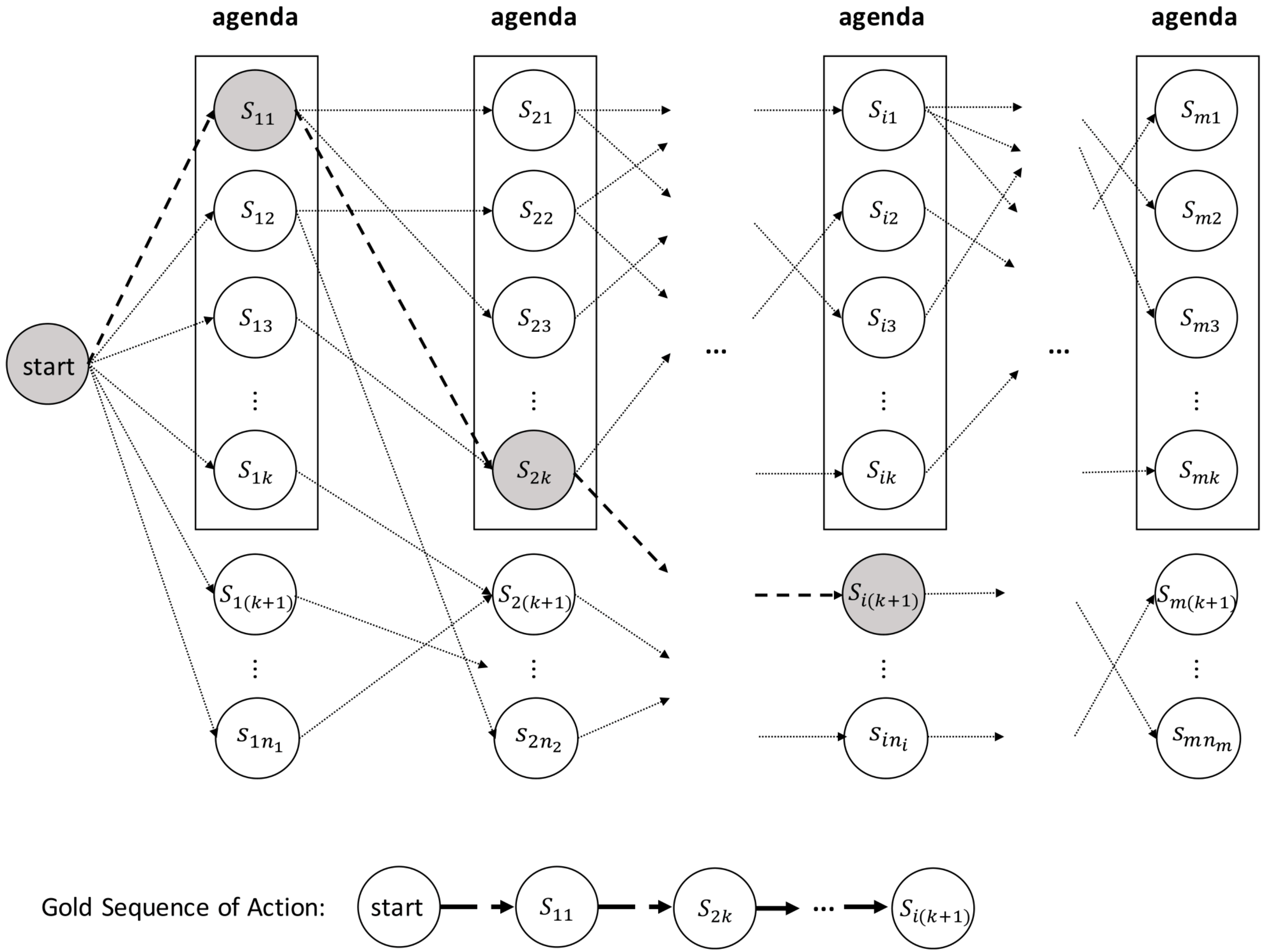
# Perceptron Training for Guiding Beam-search Decoding

WestlakeNLP

- Basic idea
  - Use the current model parameter $\vec{\theta}$ to decode training instances by beam search
  - If the model makes a mistake, update $\vec{\theta}$
- Two types of updates
  - At the $i$-th step, the gold local structure sequence $G_{1:i}$ falls out of agenda/beam
  - The highest-scored output $\widehat{T_{1:n}}$ has a higher score compared with $G_{1:n}$
- Update method
  - Standard perceptron algorithm
  - Mistake 1: $G_{1:i}$ (positive example), $\widehat{T_{1:i}}$ (negative example)
  - Mistake 2: $G_{1:n}$ (positive example), $\widehat{T_{1:n}}$ (negative example)

# Beam Search Training Algorithm

WestlakeNLP



# Beam Search Training Algorithm

WestlakeNLP

**Inputs**: $D$ — gold standard training set; $M$ — total number of training instances;
$k$ — beam size;
**Initialisation**: $\vec{\theta} \leftarrow 0$;
**Algorithm**:

```
for t ∈ [1, ..., M] do
    for (W_{1:n}, G_{1:n}) ∈ D do
        agenda ← [([], 0)]
        for i ∈ [1, ..., n] do
            candidates ← agenda;
            agenda ← [];
            for candidate ∈ candidates do
                T_{1:i-1} ← candidate[0];
                score ← candidate[1];
                for t ∈ L do
                    T_{1:i} ← EXPAND(T_{1:i-1}, t);
                    new_score ← score + θ⃗ · φ⃗_Δ(W_{1:n}, T_{1:i-1}, t);
                    APPEND(agenda, (T_{1:i}, new_score));
            agenda ← TOP-K(agenda, k);
            if not CONTAIN(G_{1:i}, agenda) then
                pos ← G_{1:i};
                neg ← TOP-K(agenda, 1)[0];
                θ⃗ ← θ⃗ + φ⃗(pos) − φ⃗(neg);
                return;
        if G_{1:n} ≠ TOP-K(agenda, 1)[0] then
            θ⃗ ← θ⃗ + φ⃗(G_{1:n}) − φ⃗(TOP-K(agenda, 1)[0]);
```

**Output**: $\vec{\theta}$;

# Contents

# Summary

WestlakeNLP

- Sequence segmentation using Sequence Labeling
- Discriminative models for directly solving sequence segmentation tasks
- Semi-Markov Conditional Random Fields
- A learning guided beam search framework using perceptron training