Dependency Parsing

lecture 4

Jakub Waszczuk, Kilian Evang

Heinrich Heine Universität

Summer semester 2021

Data-driven parsing

- Data-driven $\rightarrow$ machine learning
- Parametrize a model
- Supervised: learn parameters from annotated data
- Unsupervised: induce parameters from a large corpus
- Data-driven vs grammar-driven
    - Can parse all sentences vs. generate specific language
    - Data-driven = grammar of Σ^*

Parsing as classification

I/O

- Input: $\mathbf{x} \in \mathcal{X}$
  - e.g., document or sentence with words $\mathbf{x} = w_1 w_2 \dots w_n$ or a series or previous actions
- Output: $\mathbf{y} \in \mathcal{Y}$
  - e.g., dependency tree, document class, part-of-speech tags, next parsing actions

Parsing as classification

I/O

- Input: $\mathbf{x} \in \mathcal{X}$
  - e.g., document or sentence with words $\mathbf{x} = w_1 w_2 \dots w_n$ or a series or previous actions
- Output: $\mathbf{y} \in \mathcal{Y}$
  - e.g., dependency tree, document class, part-of-speech tags, next parsing actions

Feature representations

- We assume a mapping from $\mathbf{x}$ to a feature vector
  - $\mathbf{f}(\mathbf{x}) : \mathcal{X} \to \mathbb{R}^m$
- Or from input/output pair to a feature vector
  - $\mathbf{f}(\mathbf{x}, \mathbf{y}) : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^m$

Examples

- $\boldsymbol{x}$ is a document

$$\mathbf{f}_j(\boldsymbol{x}) = \begin{cases} 1 & \text{if } \boldsymbol{x} \text{ contains the word ``interest''} \\ 0 & \text{otherwise} \end{cases}$$

$$\mathbf{f}_j(\boldsymbol{x}) = \text{The percentage of words than contain punctuation}$$

- $\boldsymbol{x}$ is a word and $\boldsymbol{y}$ is a part-of-speech tag

$$\mathbf{f}_j(\boldsymbol{x}, \boldsymbol{y}) = \begin{cases} 1 & \text{if } \boldsymbol{x} = \text{ ``bank'' and } \boldsymbol{y} = \text{ Verb} \\ 0 & \text{otherwise} \end{cases}$$

Example 2

$$\mathbf{f}_0(\boldsymbol{x}) = \begin{cases} 1 & \text{if } \boldsymbol{x} \text{ contains the word ``John''} \\ 0 & \text{otherwise} \end{cases}$$

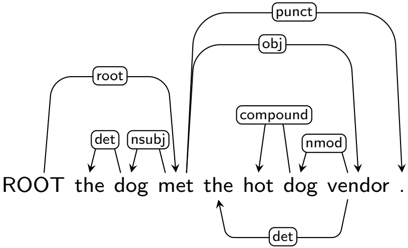
$$\mathbf{f}_1(\boldsymbol{x}) = \begin{cases} 1 & \text{if } \boldsymbol{x} \text{ contains the word ``Mary''} \\ 0 & \text{otherwise} \end{cases}$$

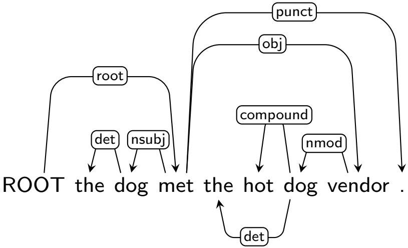
$$\mathbf{f}_2(\boldsymbol{x}) = \begin{cases} 1 & \text{if } \boldsymbol{x} \text{ contains the word ``Harry''} \\ 0 & \text{otherwise} \end{cases}$$

$$\mathbf{f}_3(\boldsymbol{x}) = \begin{cases} 1 & \text{if } \boldsymbol{x} \text{ contains the word ``likes''} \\ 0 & \text{otherwise} \end{cases}$$

- $\boldsymbol{x}$=John likes Mary $\rightarrow \mathbf{f}(\boldsymbol{x}) = [1\ 1\ 0\ 1]$
- $\boldsymbol{x}$=Mary likes John $\rightarrow \mathbf{f}(\boldsymbol{x}) = [1\ 1\ 0\ 1]$
- $\boldsymbol{x}$=Harry likes Mary $\rightarrow \mathbf{f}(\boldsymbol{x}) = [0\ 1\ 1\ 1]$
- $\boldsymbol{x}$=Harry likes Harry $\rightarrow \mathbf{f}(\boldsymbol{x}) = [0\ 0\ 1\ 1]$

Example 3





$$\mathbf{f}_j(\langle w_1, \ldots, w_n \rangle, A) = \begin{cases} 1 & \exists_{i,j,k}\ (i, j, \text{nmod}) \in A \wedge (j, k, \text{det}) \in A \\ 0 & \text{otherwise} \end{cases}$$

Linear Classifiers

- **Linear classifier**: score (or probability) of a particular classification is based on a linear combination of features and their weights
- Let $\mathbf{w} \in \mathbb{R}^m$ be a high dimensional weight vector
- If we assume that $\mathbf{w}$ is known, then we can define two kinds of linear classifiers
  - Reminder:

$$\mathbf{v} \cdot \mathbf{v}' = \sum_j \mathbf{v}_j \times \mathbf{v}'_j \in \mathbb{R}$$

  - **Binary Classification**: $\mathcal{Y} = \{-1, 1\}$

$$\boldsymbol{y} = sign(\mathbf{w} \cdot \mathbf{f}(\boldsymbol{x}))$$

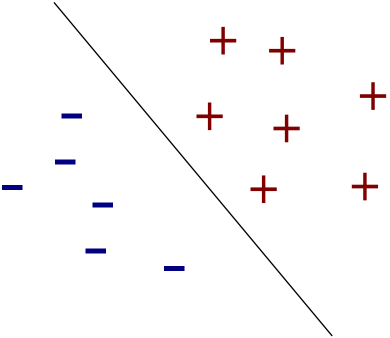
  - **Multiclass Classification**: $\mathcal{Y} = \{0, 1, \ldots, N\}$

$$\boldsymbol{y} = \arg\max_{\boldsymbol{y}} \; \mathbf{w} \cdot \mathbf{f}(\boldsymbol{x}, \boldsymbol{y})$$

Binary Linear Classifier

Divides all points:

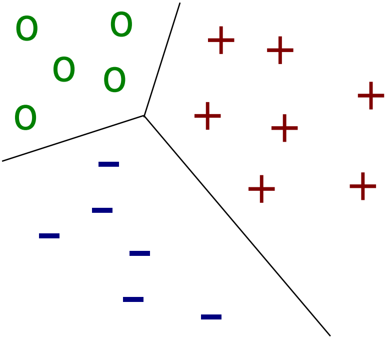
$$y = sign(\mathbf{w} \cdot \mathbf{f}(x))$$



Multiclass Linear Classifier

Defines regions of space:

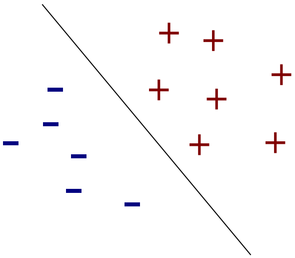
$$\boldsymbol{y} = \arg\max_{\boldsymbol{y}} \ \mathbf{w} \cdot \mathbf{f}(\boldsymbol{x}, \boldsymbol{y})$$



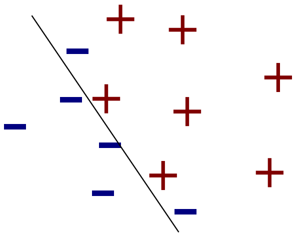
Separability

- A set of points is separable, if there exists a **w** such that classification is perfect

Separable



Not Separable



Supervised learning

- Input: training examples $\mathcal{T} = \{(\mathbf{x}_t, \mathbf{y}_t)\}_{t=1}^{|\mathcal{T}|}$
- Input: feature representation $\mathbf{f}$
- Output: feature weights $\mathbf{w}$ that maximize/minimize an objective function

Supervised learning more generally

- Input: training examples $\mathcal{T} = \{(\mathbf{x}_t, \mathbf{y}_t)\}_{t=1}^{|\mathcal{T}|}$
- Input: ~~feature representation **f**~~ model **f** parametrized with **w**
- Output: ~~feature~~ model weights **w** that maximize/minimize an objective function

Structured prediction: Issue

- Sometimes $\mathcal{Y}$ is not simply a category
- Examples (given sentence $\mathbf{x}$):
  - **Parsing**: $\mathcal{Y}$ is the set of possible trees
  - **Sequence tagging**: $\mathcal{Y}$ is the set of possible tag sequences, e.g., part-of-speech tags, named-entity tags
  - **Machine translation**: $\mathcal{Y}$ is the set of possible target language sentences
- Can't we just use our multi-class learning algorithms?
- In all the cases, the size of the set $\mathcal{Y}$ is exponential in the length of the input $\mathbf{x}$

Structured prediction: Solutions

- Sequence labeling
  - Dependency structure ⇔ fixed-length sequence of labels
- Seq2seq parsing
  - Dependency structure ⇔ variable-length sequence of labels
- Transition-based parsing
  - Dependency structure ⇔ sequence of transitions
- Graph-based parsing
  - Feature factorization, specialized decoding algorithms

Feature factorization: CFG example

$$\mathbf{f}(\mathbf{x}, \mathbf{y}) = \sum_{A \to \alpha \in \mathbf{y}} \mathbf{f}(\mathbf{x}, A \to \alpha)$$

In words: the vector of features for a given (input sentence, output derivation) pair $(\mathbf{x}, \mathbf{y})$ is the (element-wise) sum of feature vectors of all instances of production rules used in $\mathbf{y}$ within the context of $\mathbf{x}$.

Feature factorization: Bilexical CFG example

Each of the following is a feature vector corresponding to a particular binary feature:

- $\textsc{head}(w)$: word w is a head
- $\textsc{root}(w)$: word w is a root
- $\textsc{left}(v, w)$: word w is a left dependent of word v
- $\textsc{pdep}(p, q)$: a word PoS-tagged with q is a dependent of a word tagged with p
- $\textsc{betw}_1(r)$: a word tagged with r is between the head and dependent of an arc
- $\textsc{betw}(p, r, q) = \textsc{pdep}(p, q)$ & $\textsc{betw}_1(r)$: a word tagged with q is a dependent of a word tagged p and there is a word tagged with r in between

Feature factorization: Bilexical CFG example

Feature function

$$\mathbf{f}(\mathbf{x}, \mathsf{ROOT} \to \mathsf{Xw}_i) = \textsc{root}(w_i) \qquad \text{root dependency}$$

$$\mathbf{f}(\mathbf{x}, \mathsf{Xw}_i \to \mathsf{Xw}_i\ \mathsf{Xw}_j) = \textsc{right}(w_i, w_j) + \textsc{pdep}(p_i, p_j) + \sum_{k:\ i<k<j} \textsc{betw}(p_i, p_k, p_j) \qquad \text{left dependency}$$

$$\mathbf{f}(\mathbf{x}, \mathsf{Xw}_i \to \mathsf{Xw}_j\ \mathsf{Xw}_i) = \textsc{left}(w_i, w_j) + \textsc{pdep}(p_i, p_j) + \sum_{k:\ j<k<i} \textsc{betw}(p_i, p_k, p_j) \qquad \text{left dependency}$$

$$\mathbf{f}(\mathbf{x}, \mathsf{Xw}_i \to w_i) = 0 \qquad \text{terminal dependency}$$

where $\mathbf{x} = w_1 \dots w_n$ and p_i is the PoS tag assigned to word w_i

Feature factorization: Bilexical CFG example

Feature function

$$\mathbf{f}(\mathbf{x}, \mathsf{ROOT} \rightarrow \mathsf{Xw}_i) = \textsc{root}(w_i) \qquad \text{root dependency}$$

$$\mathbf{f}(\mathbf{x}, \mathsf{Xw}_i \rightarrow \mathsf{Xw}_i\ \mathsf{Xw}_j) = \textsc{right}(w_i, w_j) + \textsc{pdep}(p_i, p_j) + \sum_{k:\ i<k<j} \textsc{betw}(p_i, p_k, p_j) \qquad \text{left dependency}$$

$$\mathbf{f}(\mathbf{x}, \mathsf{Xw}_i \rightarrow \mathsf{Xw}_j\ \mathsf{Xw}_i) = \textsc{left}(w_i, w_j) + \textsc{pdep}(p_i, p_j) + \sum_{k:\ j<k<i} \textsc{betw}(p_i, p_k, p_j) \qquad \text{left dependency}$$

$$\mathbf{f}(\mathbf{x}, \mathsf{Xw}_i \rightarrow w_i) = 0 \qquad \text{terminal dependency}$$

where $\mathbf{x} = w_1 \dots w_n$ and p_i is the PoS tag assigned to word w_i (**warning**: based on the assumption we have the PoS tags on input!)

Feature factorization: Bilexical CFG example

Grammar

$\Sigma = \{\text{the}, \text{hot}, \text{dog}, \text{vendor}\}$ and Π :

- ROOT $\rightarrow$ Xvendor
- Xvendor $\rightarrow$ Xdog Xvendor | vendor
- Xhot $\rightarrow$ hot
- Xdog $\rightarrow$ Xthe Xdog | dog
- Xthe $\rightarrow$ the

the hot dog vendor

Features

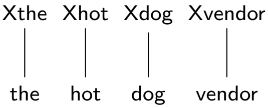
Feature factorization: Bilexical CFG example

Grammar

$\Sigma = \{\text{the}, \text{hot}, \text{dog}, \text{vendor}\}$ and Π :

- ROOT $\rightarrow$ Xvendor
- Xvendor $\rightarrow$ Xdog Xvendor | vendor
- Xhot $\rightarrow$ hot
- Xdog $\rightarrow$ Xthe Xdog | dog
- Xthe $\rightarrow$ the

Features



Feature factorization: Bilexical CFG example

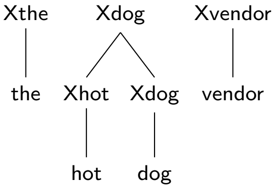
Grammar

$\Sigma = \{\text{the}, \text{hot}, \text{dog}, \text{vendor}\}$ and Π :

- ROOT $\rightarrow$ Xvendor
- Xvendor $\rightarrow$ Xdog Xvendor | vendor
- Xhot $\rightarrow$ hot
- Xdog $\rightarrow$ Xthe Xdog | dog
- Xthe $\rightarrow$ the

Features

- $\textsc{left}(\text{dog}, \text{hot}) + \textsc{pdep}(\text{noun}, \text{adj})$



Feature factorization: Bilexical CFG example

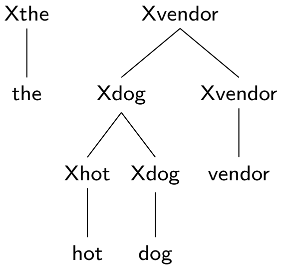
Grammar

$\Sigma = \{\text{the}, \text{hot}, \text{dog}, \text{vendor}\}$ and Π :

- ROOT $\rightarrow$ Xvendor
- Xvendor $\rightarrow$ Xdog Xvendor | vendor
- Xhot $\rightarrow$ hot
- Xdog $\rightarrow$ Xthe Xdog | dog
- Xthe $\rightarrow$ the

Features

- $\textsc{left}(\text{dog}, \text{hot}) + \textsc{pdep}(\text{noun}, \text{adj})$
- $\textsc{left}(\text{vendor}, \text{dog}) + \textsc{pdep}(\text{noun}, \text{noun})$



Feature factorization: Bilexical CFG example

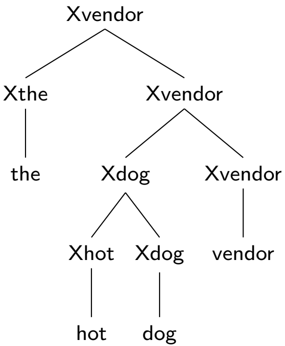
Grammar

$\Sigma = \{\text{the}, \text{hot}, \text{dog}, \text{vendor}\}$ and Π :

- ROOT → Xvendor
- Xvendor → Xdog Xvendor | vendor
- Xhot → hot
- Xdog → Xthe Xdog | dog
- Xthe → the

Features

- $\textsc{left}(\text{dog}, \text{hot}) + \textsc{pdep}(\text{noun}, \text{adj})$
- $\textsc{left}(\text{vendor}, \text{dog}) + \textsc{pdep}(\text{noun}, \text{noun})$
- $\textsc{left}(\text{vendor}, \text{the}) + \textsc{pdep}(\text{noun}, \text{det}) + \textsc{betw}(\text{noun}, \text{noun}, \text{det}) + \textsc{betw}(\text{noun}, \text{adj}, \text{det})$



Feature factorization: Bilexical CFG example

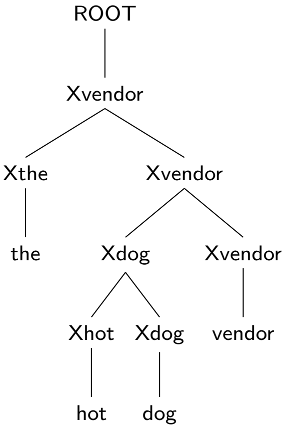
Grammar

$\Sigma = \{\text{the}, \text{hot}, \text{dog}, \text{vendor}\}$ and Π :

- ROOT $\rightarrow$ Xvendor
- Xvendor $\rightarrow$ Xdog Xvendor | vendor
- Xhot $\rightarrow$ hot
- Xdog $\rightarrow$ Xthe Xdog | dog
- Xthe $\rightarrow$ the

Features

- $\textsc{left}(\text{dog}, \text{hot}) + \textsc{pdep}(\text{noun}, \text{adj})$
- $\textsc{left}(\text{vendor}, \text{dog}) + \textsc{pdep}(\text{noun}, \text{noun})$
- $\textsc{left}(\text{vendor}, \text{the}) + \textsc{pdep}(\text{noun}, \text{det}) +$ $\textsc{betw}(\text{noun}, \text{noun}, \text{det}) + \textsc{betw}(\text{noun}, \text{adj}, \text{det})$
- $\textsc{root}(\text{vendor})$



Feature factorization: Bilexical CFG example

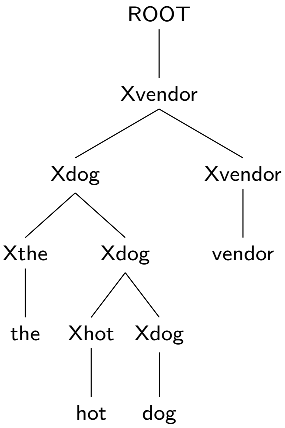
Grammar

$\Sigma = \{\text{the}, \text{hot}, \text{dog}, \text{vendor}\}$ and Π :

- ROOT $\rightarrow$ Xvendor
- Xvendor $\rightarrow$ Xdog Xvendor | vendor
- Xhot $\rightarrow$ hot
- Xdog $\rightarrow$ Xthe Xdog | dog
- Xthe $\rightarrow$ the

Features

- LEFT(dog, hot) + PDEP(noun, adj)
- LEFT(vendor, dog) + PDEP(noun, noun)
- LEFT(dog, the) + PDEP(noun, det) + BETW(noun, adj, det)
- ROOT(vendor)



Feature factorization: Bilexical CFG example

Dense feature representations

$$\mathbf{f}(\mathbf{x}, \text{ROOT} \rightarrow \text{Xw}_i) = W^{(0)} x_i + b^{(0)} \qquad \text{root dependency}$$

$$\mathbf{f}(\mathbf{x}, \text{Xw}_i \rightarrow \text{Xw}_i \ \text{Xw}_j) = W^{(r)} [x_i; x_j] + b^{(r)} \qquad \text{right dependency}$$

$$\mathbf{f}(\mathbf{x}, \text{Xw}_i \rightarrow \text{Xw}_j \ \text{Xw}_i) = W^{(l)} [x_i; x_j] + b^{(l)} \qquad \text{left dependency}$$

$$\mathbf{f}(\mathbf{x}, \text{Xw}_i \rightarrow w_i) = 0 \qquad \text{terminal dependency}$$

where

- w_i is represented by the corresponding word embedding vector x_i
- $W^{(0)}$, $W^{(l)}$, $W^{(l)}$, $b^{(0)}$, $b^{(l)}$, $b^{(l)}$ are parameters of the model
- $[x; y]$ denotes vector concatenation of x and y

Feature representation in the era of deep learning

- Contextualized vector-based representations of words
- Minimalistic feature functions
- Feature discovery/extraction is part of the model
- The rest is (roughly) the same

Feature representation in the era of deep learning

- Contextualized vector-based representations of words
- Minimalistic feature functions
- Feature discovery/extraction is part of the model
- The rest is (roughly) the same

$\Rightarrow$ we will often rely on manual feature representations and linear scoring for the sake of simplicity and transparency

T H E

E N D

Bonus: Weighted CYK

Input

- Bilexical CFG (N, Σ, Π, S)
- Input sentence $w_1 w_2 \dots w_n \in \Sigma^*$ of length n
- NEW: Feature function $\mathbf{f}$ factored relative to production rules
- NEW: Weight vector $\mathbf{w}$

Item

- Each item has the form $[A, i, j] : s$ where $A \in N$, $1 \leq i \leq j \leq n$, and (NEW:) $s \in \mathbb{R}$ is the corresponding score
- Item $[A, i, j] : s$ states that $A \Rightarrow^* w_i \dots w_j$ and (NEW:) the (minimal) score of the corresponding derivation is s

Bonus: Weighted CYK

Rules

- Axiom + terminal dependency score:

$$\frac{}{[A, i, i]} \quad A \rightarrow w_i \in \Pi$$

- Combine + right dependency score:

$$\frac{[A, i, j] \qquad [B, j+1, k]}{[A, i, k]} \quad A \rightarrow AB \in \Pi$$

- Root + root dependency score:

$$\frac{[A, 1, n]}{[\mathrm{ROOT}, 1, n]} \quad \mathrm{ROOT} \rightarrow A \in \Pi$$

Bonus: Weighted CYK

Rules

- Axiom + terminal dependency score:

$$\frac{}{[A, i, i] : \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, A \rightarrow w_i)} \quad A \rightarrow w_i \in \Pi$$

- Combine + right dependency score:

$$\frac{[A, i, j] \qquad [B, j+1, k]}{[A, i, k]} \quad A \rightarrow AB \in \Pi$$

- Root + root dependency score:

$$\frac{[A, 1, n]}{[\mathrm{ROOT}, 1, n]} \quad \mathrm{ROOT} \rightarrow A \in \Pi$$

Bonus: Weighted CYK

Rules

- Axiom + terminal dependency score:

$$\frac{}{[A, i, i] : \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, A \rightarrow w_i)} \quad A \rightarrow w_i \in \Pi$$

- Combine + right dependency score:

$$\frac{[A, i, j] : s_1 \quad [B, j+1, k] : s_2}{[A, i, k] : s_1 + s_2 + \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, A \rightarrow AB)} \quad A \rightarrow AB \in \Pi$$

- Root + root dependency score:

$$\frac{[A, 1, n]}{[\text{ROOT}, 1, n]} \quad \text{ROOT} \rightarrow A \in \Pi$$

Bonus: Weighted CYK

Rules

- Axiom + terminal dependency score:

$$\frac{}{[A, i, i] : \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, A \to w_i)} \quad A \to w_i \in \Pi$$

- Combine + right dependency score:

$$\frac{[A, i, j] : s_1 \quad [B, j+1, k] : s_2}{[A, i, k] : s_1 + s_2 + \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, A \to AB)} \quad A \to AB \in \Pi$$

- Root + root dependency score:

$$\frac{[A, 1, n] : s}{[\text{ROOT}, 1, n] : s + \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, \text{ROOT} \to A)} \quad \text{ROOT} \to A \in \Pi$$

Bonus: Weighted CYK

This works thanks to:

$$\mathbf{w} \cdot \mathbf{f}(\mathbf{x}, \mathbf{y}) = \mathbf{w} \cdot \sum_{A \rightarrow \alpha \in \mathbf{y}} \mathbf{f}(\mathbf{x}, A \rightarrow \alpha) = \sum_{A \rightarrow \alpha \in \mathbf{y}} \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, A \rightarrow \alpha)$$