Transition-based parsing: Learning

Goal: Given an input sentence, obtain its dependency graph.

We assume we have the dependency graph for many sentences.

Previous lecture: given a sentence and its dependency graph, how we can get the sequence of transitions.
Recap: A configuration is {a stack, a buffer, a set of Arcs}; a transition takes one configuration to another

So, we have training data {configuration Ci, transition t} giving what transition t can be followed from which configuration Ci

Now - how to use this data to train a (classifier) model to parse a new sentence (i.e., predict which transition to use for a given configuration, to finally obtain the dependency graph of the new sentence)? [a 4-class classification problem]

Classifier-Based Parsing

Data-driven deterministic parsing:

- Deterministic parsing requires an **oracle**.
- An oracle can be approximated by a **classifier**.
- A classifier can be trained using **treebank** data.

Oracle: an entity that can say what transition to take from which configuration

Treebank data - corpus of sentences and their dependency graphs (from where we can obtain the configurations and the transitions taken from those configurations)

The Oracle is developed from the Treebank data.

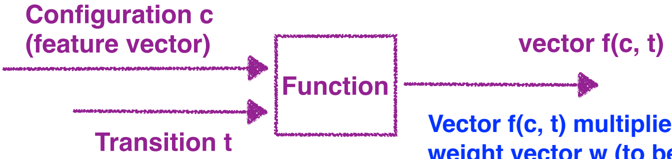
Classifier-Based Parsing

Data-driven deterministic parsing:

- Deterministic parsing requires an **oracle**.
- An oracle can be approximated by a **classifier**.
- A classifier can be trained using **treebank** data.

Learning Problem

Approximate a function from **configurations**, represented by feature vectors to **transitions**, given a training set of gold standard **transition sequences**.



Vector f(c, t) multiplied by a weight vector w (to be learned) will give likelihood of t for c

Classifier-Based Parsing

Data-driven deterministic parsing:

- Deterministic parsing requires an **oracle**.
- An oracle can be approximated by a **classifier**.
- A classifier can be trained using **treebank** data.

Learning Problem

Approximate a function from **configurations**, represented by feature vectors to **transitions**, given a training set of gold standard **transition sequences**.

Three issues

- How to represent configurations by feature vectors?
- How to derive training data from treebanks?
- How to learn classifiers?

Feature Models

Configuration C = {Stack S, Buffer B, set of Arcs G}

A feature representation $f(c)$ of a configuration c is a vector of simple features $f_i(c)$.

Typical Features

- Nodes:
  - Target nodes (top of S, head of B)
  - Linear context (neighbors in S and B)
  - Structural context (parents, children, siblings in G)
- Attributes:
  - Word form (and lemma)
  - Part-of-speech (and morpho-syntactic features)
  - Dependency type (if labeled)
  - Distance (between target tokens)

Structural context: the arcs that have already been identified

Deterministic Parsing

To guide the parser, a linear classifier can be used:

$$t^* = \arg\max_t w.f(c,t)$$

Weight vector w learned from treebank data.

4-class classifier - predicts one of the 4 transitions for the given configuration

t* is that transition that maximizes w . f(c, t)

Using classifier at run-time

PARSE($w_1, \ldots, w_n$)

1 $c \leftarrow ([]_S, [w_1, \ldots, w_n]_B, \{\})$ **Initial configuration**

2 **while** $B_c \neq []$ **while buffer is not empty**

3 $t^* \leftarrow \arg\max_t w.f(c,t)$

4 $c \leftarrow t^*(c)$

5 **return** $T = (\{w_1, \ldots, w_n\}, A_c)$

How to learn the weight vector w? We need training data (from Treebank)

Training data

- Training instances have the form $(f(c), t)$, where
  - $f(c)$ is a feature representation of a configuration c,
  - t is the correct transition out of c (i.e., $o(c) = t$).

Training data

- Training instances have the form $(f(c), t)$, where
  - $f(c)$ is a feature representation of a configuration c,
  - t is the correct transition out of c (i.e., $o(c) = t$).
- Given a dependency treebank, we can sample the oracle function o as follows:
  - For each sentence x with gold standard dependency graph G_x, construct a transition sequence $C_{0,m} = (c_0, c_1, \ldots, c_m)$ such that

    $$c_0 = c_s(x), \quad G_{c_m} = G_x$$

    This is what we discussed in the previous lecture
  - For each configuration $c_i (i < m)$, we construct a training instance $(f(c_i), t_i)$, where $t_i(c_i) = c_{i+1}$.

Standard Oracle for Arc-Eager Parsing

$o(c,T) =$

- **Left-Arc** if top(S_c) $\leftarrow$ first(B_c) in T
- **Right-Arc** if top(S_c) $\rightarrow$ first(B_c) in T
- **Reduce** if $\exists w < top(S_c)$: $w \leftrightarrow$ first(B_c) in T
- **Shift** otherwise

This is exactly what we followed in the previous lecture

Online Learning with an Oracle

LEARN($\{T_1, \ldots, T_N\}$)

$w \leftarrow 0.0$ **Initialize classifier weights (with any real value)**

for i in $1..K$ **K: number of iterations over training data**

  for j in $1..N$ **N: number of instances in training data**

    $c \leftarrow ([]_S, [w_1, \ldots, w_{n_j}]_B, \{\})$

    while $B_c \neq []$

      $t^* \leftarrow \arg\max_t w.f(c,t)$ **t* prediction by present classifier (may be with sub-optimal weights)**

      $t_o \leftarrow o(c, T_i)$ **actual transition according to oracle**

      if $t^* \neq t_o$

        $w \leftarrow w + f(c,t_o) - f(c,t^*)$ **update weights towards the actual transition, away from the wrongly predicted transition**

      $c \leftarrow t_o(c)$

return w

Oracle $o(c, T_i)$ returns the optimal transition of c and T_i

Example

Consider the sentence, ‘John saw Mary’.

- Draw a dependency graph for this sentence.
- Assume that you are learning a classifier for the data-driven deterministic parsing and the above sentence is a gold-standard parse in your training data. You are also given that *John* and *Mary* are ‘Nouns’, while the POS tag of *saw* is ‘Verb’. Assume that your features correspond to the following conditions:
  - The stack is empty **C1**
  - Top of stack is Noun and Top of buffer is Verb **C2**
  - Top of stack is Verb and Top of buffer is Noun **C3**

  Initialize the weights of all your features to *5.0*, except that in all of the above cases, you give a weight of *5.5* to *Left-Arc*. Define your feature vector and the initial weight vector.
- Use this gold standard parse during online learning and report the weights after completing one full iteration of Arc-Eager parsing over this sentence.