Learning From Large Codebases: Chapter 1 (part 2)

Prediction algorithm

$$\boldsymbol{y} = \underset{\boldsymbol{y}' \in \Omega_x}{\arg\max}\, Pr(\boldsymbol{y}' | x) = \underset{\boldsymbol{y}' \in \Omega_x}{\arg\max}\, score(\boldsymbol{y}', x) = \underset{\boldsymbol{y}' \in \Omega_x}{\arg\max}\, \boldsymbol{w}^T \boldsymbol{f}(\boldsymbol{y}', x)$$

x – input program

y – predicted labels

Prediction algorithm

for pass $\in [1 \ldots (\text{num passes})]$ **do**
    for each node v with unknown property **do**
        $E_v \leftarrow$ all edges adjacent to v
        $\text{score}_v \leftarrow \text{score}(E_v, (y, z))$
        for $l' \in \text{candidates}(v, (y, z), E_v)$ **do**
            $l \leftarrow y_v$
            $y_v \leftarrow l'$
            if $\text{score}(E_v, (y, z)) > score_v$ **and** $y \in \Omega_x$ **then**
                $\text{score}_v \leftarrow \text{score}(E_v, (y, z))$
            else
                $y_v \leftarrow l$

Prediction algorithm

$$\mathrm{score}(E, A) = \sum_{(a,b,rel)\in E} \sum_{i=1}^{k} w_i \psi_i(A_a, A_b, rel)$$

$$\begin{aligned} &\mathit{candidates}(v, A, E) = \\ &= \bigcup_{\langle a,v,rel\rangle \in E} \{l^2 \mid \langle l^1, l^2, r\rangle \in \mathit{topL}_s(A_a, rel)\} \quad \cup \\ &\quad \bigcup_{\langle v,b,rel\rangle \in E} \{l^1 \mid \langle l^1, l^2, r\rangle \in \mathit{topR}_s(A_b, rel)\} \end{aligned}$$

Additional improvements

- Control number of candidates
- Optimization on pairs of nodes

Learning parameters

Our goal:

$$\forall j, \forall \boldsymbol{y}' \in \Omega_{x^{(j)}} \;\; score(\boldsymbol{y}^{(j)}, x^{(j)}) \geq score(\boldsymbol{y}', x^{(j)}) \; + \; \Delta(\boldsymbol{y}^{(j)}, \boldsymbol{y}')$$

For every object $(x^{(j)}, y^{(j)})$ in training set.

$\Delta(y^{(j)}, y')$ – margin function

Loss function

Loss function for one object:

$$\ell(\boldsymbol{w}; x^{(j)}, \boldsymbol{y}^{(j)}) = \max_{\boldsymbol{y}' \in \Omega_{x^{(j)}}} \left(\boldsymbol{w}^T [\boldsymbol{f}(\boldsymbol{y}', x^{(j)}) - \boldsymbol{f}(\boldsymbol{y}^{(j)}, x^{(j)})] + \Delta(\boldsymbol{y}^{(j)}, \boldsymbol{y}') \right)$$

Optimal parameters can be found as:

$$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w} \in \mathcal{W}_\lambda} \sum_{j=1}^{t} \ell(\boldsymbol{w}; x^{(j)}, \boldsymbol{y}^{(j)})$$

Projected Stochastic Gradient Descent

$W_\lambda = \{w | w_i \in [0, 1/\lambda] \text{ for all } i\}$
while w not converged **do**
  $g \leftarrow \nabla_w l(w, x^{(j)}, y^{(j)})$
  $w \leftarrow w - \alpha g$
  $w \leftarrow \mathrm{Proj}_{W_\lambda}(w)$

$$\mathrm{Proj}_{W_\lambda}(w) : w'_i = \max(0, \min(1/\lambda, w_i))$$

How to compute gradients?

$$g = f(y_{\text{best}}, x^{(j)}) - f(y^{(j)}, x^{(j)})$$

$$y_{\text{best}} = \arg \max_{y' \in \Omega_{x^{(j)}}} (\text{score}(y', x^{(j)}) + \Delta(y^{(j)}, y'))$$

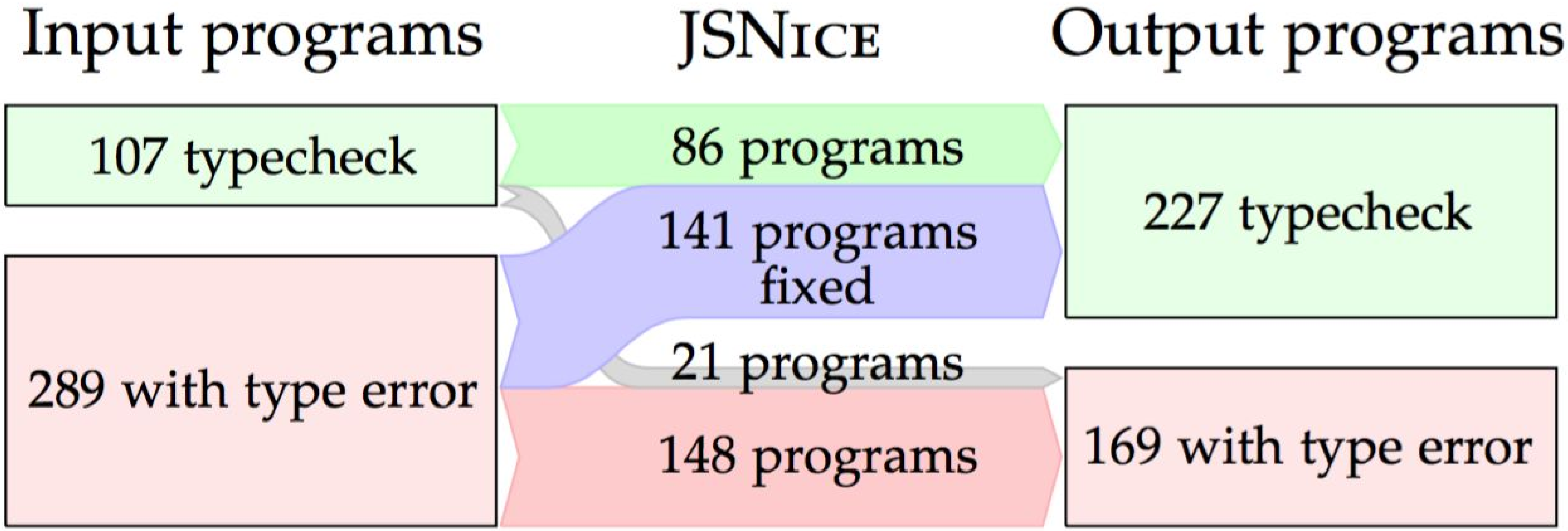
Details

- Lambda and delta are chosen via cross-validation.
- Train set consists of 324,501 files. Test set consists of 2,710 files.
- Train set is collected from GitHub, test set is collected from BitBucket

Results

System	Names Accuracy	Types Precision	Types Recall
all training data	**63.4%**	**81.6%**	**66.9%**
10% of training data	54.5%	81.4%	64.8%
1% of training data	41.2%	77.9%	62.8%
all data, no structure	54.1%	84.0%	56.0%
baseline - no predictions	25.3%	37.8%	100%

Typechecking Results



Running times

Beam size parameter	**Name prediction**		**Type prediction**	
b	Accuracy	Time	Precision	Time
4	57.9%	43ms	80.6%	36ms
8	59.2%	60ms	80.9%	39ms
16	62.8%	62ms	81.6%	33ms
32	63.2%	80ms	81.3%	37ms
64 **(JSNice)**	63.4%	114ms	81.6%	40ms
128	63.5%	175ms	82.0%	42ms
256	63.5%	275ms	81.6%	50ms
Naïve greedy, no beam	62.8%	115.2 s	81.7%	410ms