Prof. Ryan Cotterell

Course Assignment Episode 2

January 11, 2022

Deheng Zhang
nethz Username: dezhang
Student ID: 21-950-787

Collaborators:
Junling Wang
Tianyi Liu
Kailin Liu

By submitting this work, I verify that it is my own. That is, I have written my own solutions to each problem for which I am submitting an answer. I have listed above all others with whom I have discussed these answers.

Question 1: CFG Refinement

(a) The CFG is defined as following

(i) $\mathcal{N} \rightarrow \{\mathbf{NP}, \mathbf{Det}, \mathbf{N}, \mathbf{VP}, \mathbf{V}, \mathbf{PP}, \mathbf{P}, \mathbf{Adj}\}$

(ii) $\mathcal{S} \rightarrow \{\mathbf{S}\}$

(iii) $\mathbf{\Sigma} \rightarrow \{the, man, made, a, pot, with, clay, girl, saw, telescope, handle, blue, jeans\}$

(iv) $\mathcal{R} \rightarrow Production\ rules:$

$$
\begin{aligned}
&\mathbf{S} \rightarrow \mathbf{NP\ VP} \\
&\mathbf{NP} \rightarrow \mathbf{Det\ N} \mid \mathbf{NP\ PP} \mid \mathbf{Adj\ N} \\
&\mathbf{VP} \rightarrow \mathbf{VP\ PP} \mid \mathbf{V\ NP} \\
&\mathbf{PP} \rightarrow \mathbf{P\ N} \mid \mathbf{P\ NP} \\
&\mathbf{N} \rightarrow man \mid girl \mid pot \mid clay \mid telescope \mid handle \mid jeans \\
&\mathbf{Det} \rightarrow the \mid a \\
&\mathbf{V} \rightarrow made \mid saw \\
&\mathbf{P} \rightarrow with \\
&\mathbf{Adj} \rightarrow blue
\end{aligned}
$$

(b) The probabilities are:

$$
\begin{aligned}
&\mathbf{S} \rightarrow \mathbf{NP\ VP\ (1)} \\
&\mathbf{NP} \rightarrow \mathbf{Det\ N}\ (\mathbf{\tfrac{10}{13}}) \mid \mathbf{NP\ PP}\ (\mathbf{\tfrac{2}{13}}) \mid \mathbf{Adj\ N}(\mathbf{\tfrac{1}{13}}) \\
&\mathbf{VP} \rightarrow \mathbf{VP\ PP}\ (\mathbf{\tfrac{1}{3}}) \mid \mathbf{V\ NP}\ (\mathbf{\tfrac{2}{3}}) \\
&\mathbf{PP} \rightarrow \mathbf{P\ N}\ (\mathbf{\tfrac{1}{4}}) \mid \mathbf{P\ NP}\ (\mathbf{\tfrac{3}{4}}) \\
&\mathbf{N} \rightarrow man\ (\tfrac{1}{3}) \mid girl\ (\tfrac{1}{6}) \mid pot\ (\tfrac{1}{6}) \mid clay\ (\tfrac{1}{12}) \mid telescope\ (\tfrac{1}{12}) \mid handle\ (\tfrac{1}{12}) \mid jeans\ (\tfrac{1}{12}) \\
&\mathbf{Det} \rightarrow the\ (\tfrac{3}{5}) \mid a\ (\tfrac{2}{5}) \\
&\mathbf{V} \rightarrow made\ (\tfrac{1}{2}) \mid saw\ (\tfrac{1}{2}) \\
&\mathbf{P} \rightarrow with\ (1) \\
&\mathbf{Adj} \rightarrow blue\ (1)
\end{aligned}
$$

(c) By observation, I find the parent of **NP** can determine whether it is subject or object of the sentence. Therefore, we extend the non-terminals by adding the parent information, the non-terminal becomes: $Non-terminal(Parent)$

(i) $\mathcal{N} \rightarrow \{\mathbf{NP(S)}, \mathbf{NP(NP)}, \mathbf{NP(VP)}, \mathbf{NP(PP)}, \mathbf{Det(NP)}, \mathbf{N(NP)}, \mathbf{N(PP)}, \mathbf{VP(S)}, \mathbf{VP(VP)}, \mathbf{V(VP)}, \mathbf{PP(NP)}, \mathbf{PP(VP)}, \mathbf{P(PP)}, \mathbf{Adj(NP)}\}$

(ii) $\mathcal{S} \rightarrow \{\mathbf{S}\}$

(iii) $\mathbf{\Sigma} \rightarrow \{the, man, made, a, pot, with, clay, girl, saw, telescope, handle, blue, jeans\}$

(iv) $\mathcal{R} \rightarrow Production\ rules:$

$$\mathbf{S} \rightarrow \mathbf{NP(S)\ VP(S)\ (1)}$$
$$\mathbf{NP(S)} \rightarrow \mathbf{Det(NP)\ N(NP)\ (1)}$$
$$\mathbf{NP(NP)} \rightarrow \mathbf{Det(NP)\ N(NP)\ (1)}$$
$$\mathbf{NP(VP)} \rightarrow \mathbf{Det(NP)\ N(NP)\ (\frac{1}{2})\ |\ NP(NP)\ PP(NP)\ (\frac{1}{2})}$$
$$\mathbf{NP(PP)} \rightarrow \mathbf{Det(NP)\ N(NP)\ (\frac{2}{3})\ |\ Adj(NP)\ N(NP)(\frac{1}{3})}$$
$$\mathbf{VP(S)} \rightarrow \mathbf{VP(VP)\ PP(VP)\ (\frac{1}{2})\ |\ V(VP)\ NP(VP)\ (\frac{1}{2})}$$
$$\mathbf{VP(VP)} \rightarrow \mathbf{V(VP)\ NP(VP)\ (1)}$$
$$\mathbf{PP(NP)} \rightarrow \mathbf{P(PP)\ NP(PP)\ (1)}$$
$$\mathbf{PP(VP)} \rightarrow \mathbf{P(PP)\ N(PP)\ (\frac{1}{2})\ |\ P(PP)\ NP(PP)\ (\frac{1}{2})}$$
$$\mathbf{N(NP)} \rightarrow man\ (\frac{4}{11})\ |\ girl\ (\frac{2}{11})\ |\ pot\ (\frac{2}{11})\ |\ telescope\ (\frac{1}{11})\ |\ handle\ (\frac{1}{11})\ |\ jeans\ (\frac{1}{11})$$
$$\mathbf{N(PP)} \rightarrow clay\ (1)$$
$$\mathbf{Det(NP)} \rightarrow the\ (\frac{3}{5})\ |\ a\ (\frac{2}{5})$$
$$\mathbf{V(VP)} \rightarrow made\ (\frac{1}{2})\ |\ saw\ (\frac{1}{2})$$
$$\mathbf{P(PP)} \rightarrow with\ (1)$$
$$\mathbf{Adj(NP)} \rightarrow blue\ (1)$$

(d) The parse tree with lexicalized rules is shown in Figure 1, and the production rule is shown below:

$$S(saw) \rightarrow NP(girl)\ VP(saw)$$
$$NP(girl) \rightarrow Det(the)\ N(girl)$$
$$NP(man) \rightarrow NP(man)\ PP(jeans)$$
$$NP(man) \rightarrow Det(the)\ N(man)$$
$$NP(jeans) \rightarrow Adj(blue)\ N(jeans)$$
$$VP(saw) \rightarrow V(saw)\ NP(man)$$
$$PP(jeans) \rightarrow P(with)\ NP(jeans)$$
$$N(girl) \rightarrow girl$$
$$N(man) \rightarrow man$$
$$N(jeans) \rightarrow jeans$$
$$V(saw) \rightarrow saw$$
$$P(with) \rightarrow with$$
$$Adj(blue) \rightarrow blue$$
$$Det(the) \rightarrow the$$

(e) (i) The comlexity of the CKY algorithm is:

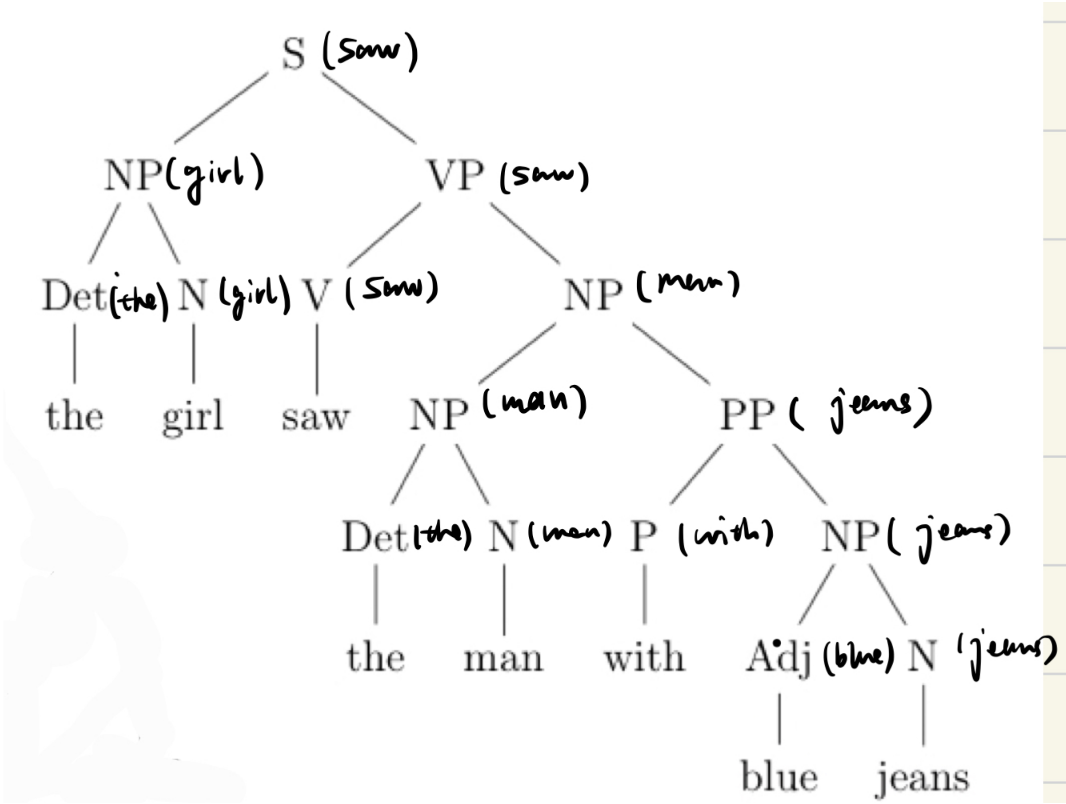

Figure 1: Parse tree of "the girl saw the man with blue jeans"

- Time: $O(N^3|\mathcal{R}|)$
- Memory: $O(N^2|\mathcal{N}|)$

, where N is the size of the sentence (number of non-terminals), $|\mathcal{R}|$ is the size of the rules set, and $|\mathcal{N}|$ is the size of the non-terminal set.

(ii) With the parent information in (c), in the worst case, the size of rule set becomes $|\mathcal{R}||\mathcal{N}|$, and the size of non-terminal set becomes $|\mathcal{N}|^2$. The complexity becomes:

- Time: $O(N^3|\mathcal{R}||\mathcal{N}|)$
- Memory: $O(N^2|\mathcal{N}|^2)$

, where N is the size of the sentence (number of non-terminals), $|\mathcal{R}|$ is the size of the previous rules set, and $|\mathcal{N}|$ is the size of the previous non-terminal set.

(iii) With the parent information in (d), in the worst case, each rule corresponds to two words (head and not head), there are $2|\Sigma|^2 - 1$ combinations. The size of rule set becomes $|\mathcal{R}|(2|\Sigma|^2 - 1)$, and the size of non-terminal set becomes $|\mathcal{N}||\Sigma|$. The complexity becomes:

- Time: $O(N^3|\mathcal{R}||\Sigma|^2)$
- Memory: $O(N^2|\mathcal{N}||\Sigma|)$

, where N is the size of the sentence (number of non-terminals), $|\mathcal{R}|$ is the size of the previous rules set, and $|\mathcal{N}|$ is the size of the previous non-terminal set.

Question 2: Parsing Projective Dependecy Trees

(a) The score of a weighted context-free grammar tree is defined as:

$$score(\mathbf{t}, w) = \sum_{r \in \mathbf{t}} score(r, w) = \sum_{(X \to Y\ Z) \in \mathbf{t}} score(X \to Y\ Z, w) + \sum_{(X \to x) \in \mathbf{t}} score(X \to x)$$

The score of of a dependency tree is defined as:

$$\begin{aligned} score(\mathbf{t}, w) &= \sum_{(i \to j) \in t} score(i, j, \mathbf{w}) + score(root = w_j, \mathbf{w}) \\ &= \sum_{(i \to j) \in t} \psi(i \to j) + \psi(ROOT \to j) \end{aligned}$$

Therefore, we define the lexicalized WCFG as following (note that N represents many different non-terminals):

(i) $\mathcal{N} \to \{N(w)\}\ (w \in \Sigma)$

(ii) $\mathcal{S} \to \{S(w)\}\ (w \in \Sigma)$

(iii) $\Sigma \to set\ of\ words$

(iv) $\mathcal{R} \to Production\ rules:$

$$\begin{aligned} &S(w_j) \to N(w_j) \\ &N(w_i) \to N(w_i)\ N(w_j) \mid N(w_j)\ N(w_i) \\ &N(w) \to w \end{aligned}$$

By observation, we can find that there is a one-to-one correspondence between the binary rules (the second rule) and the edge set of dependency trees:

$$\begin{aligned} score(\mathbf{t}, w) = \sum_{r \in \mathbf{t}} score(r, w) &= score(S(w_j) \to N(w_j), \mathbf{w}) \\ &+ \sum_{(N(w_i) \to N(w_i)\ N(w_j)) \in \mathbf{t}} score(N(w_i) \to N(w_i)\ N(w_j), \mathbf{w}) \\ &+ \sum_{(N(w) \to w) \in \mathbf{t}} score(N(w) \to w, \mathbf{w}) \end{aligned}$$

By defining the following equality, we can make sure the score of lexicalized WCFG equals to the score of corresponding dependency tree:

$$\begin{aligned} &score(S(w_j) \to N(w_j), \mathbf{w}) = \psi(ROOT \to j) \\ &score(N(w_i) \to N(w_i)\ N(w_j), \mathbf{w}) = \psi(i \to j) \\ &score(N(w) \to w, \mathbf{w}) = 0 \end{aligned}$$

(b) There are two possible lexicalized WCFG for the example

(i) The tree is shown in Figure 2.

$$score(dependency, \mathbf{w}) = \psi(we \rightarrow watch) + \psi(ROOT \rightarrow we)$$
$$score(WCFG, \mathbf{w}) = score(S(we) \rightarrow N_0(we), \mathbf{w}) + score(N_0(we) \rightarrow N_1(we)\ N_2(watch), \mathbf{w})$$
$$+ score(N_1(we) \rightarrow we + score(N_2(watch) \rightarrow watch)$$
$$= \psi(we \rightarrow watch) + \psi(ROOT \rightarrow we) + 0$$

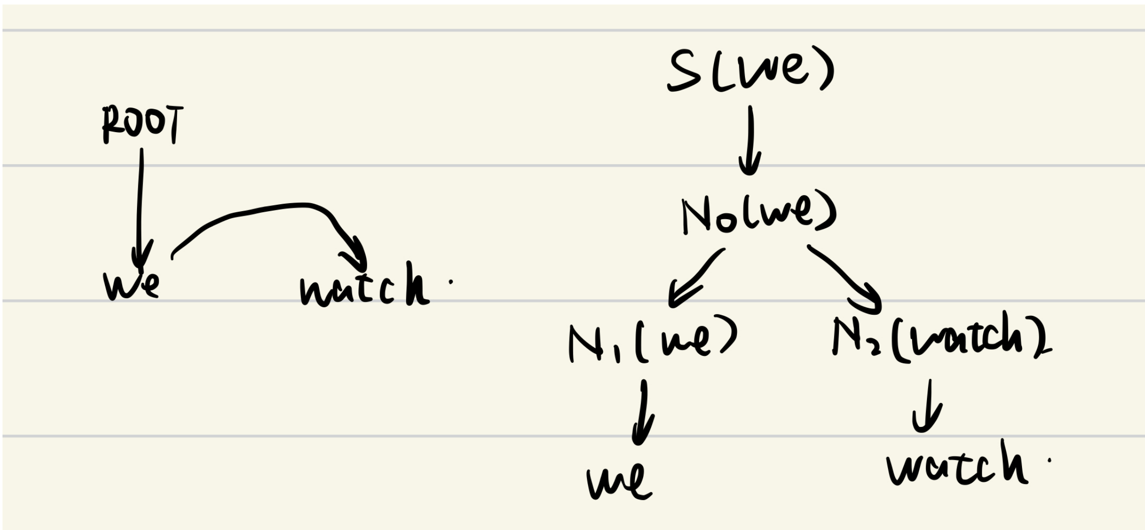

Figure 2: The first lexicalized WCFG and Dependency Tree

(ii) The tree is shown in Figure 3

$$score(dependency, \mathbf{w}) = \psi(watch \rightarrow we) + \psi(ROOT \rightarrow watch)$$
$$score(WCFG, \mathbf{w}) = score(S(watch) \rightarrow N_0(watch), \mathbf{w}) + score(N_0(watch) \rightarrow N_1(we)\ N_2(watch), \mathbf{w})$$
$$+ score(N_1(we) \rightarrow we + score(N_2(watch) \rightarrow watch)$$
$$= \psi(watch \rightarrow we) + \psi(ROOT \rightarrow watch) + 0$$

Question 3: Semantic Representations

(a) Sentences generated by ther grammar G are:

(i) everyone loves everyone

(ii) everyone loves someone

(iii) everyone loves Alex

(iv) someone loves everyone

(v) someone loves someone

(vi) someone loves Alex

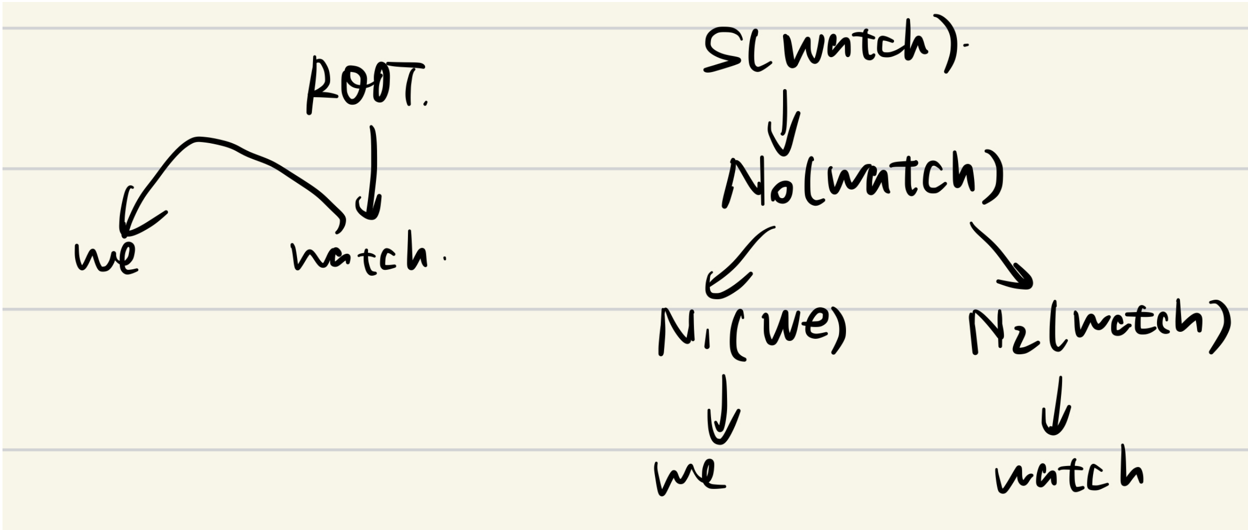

Figure 3: The second lexicalized WCFG and Dependency Tree

(vii) Alex loves everyone

(viii) Alex loves someone

(ix) Alex loves Alex

(b) The CFG tree is shown in Figure 4. The semantic representation of the terminal symbol *loves*

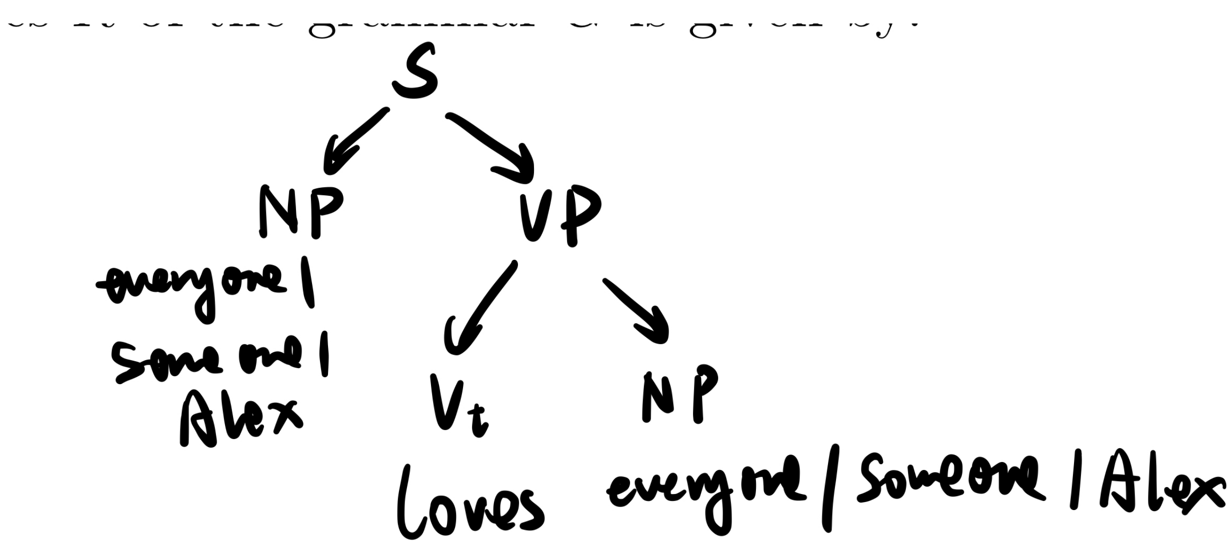

Figure 4: The CFG tree

- ”*loves*”.sem = $\lambda P.\lambda x.P(\lambda y.\text{LOVES}(x, y))$
- ”*everyone*”.sem = $\lambda P.\forall x.(\text{PERSON}(x) \implies P(x))$
- ”*someone*”.sem = $\lambda P.\exists y.(\text{PERSON}(y) \wedge P(y))$

- "*Alex*".sem = $\lambda P.P(\text{ALEX})$

In this way:
"*loves Alex*".sem = ("*loves*".sem, "*Alex*".sem)
$=(\lambda P.\lambda x.P(\lambda y.\text{LOVES}(x, y))\ \lambda P.P(\text{ALEX}))$
$=\lambda x.(\lambda P.P(\text{ALEX})\lambda y.\text{LOVES}(x, y))$
$= \lambda x.(\lambda y.\text{LOVES}(x, y)\ \text{ALEX})$
$= \lambda x.\text{LOVES}(x,\text{ALEX})$

"*Alex loves Alex*".sem = ("*Alex*".sem, "*loves Alex*".sem)
$=\lambda P.P(\text{ALEX}))\ (\ \lambda x.\text{LOVES}(x,\text{ALEX}))$
$=\text{LOVES}(\text{ALEX},\text{ALEX})$

"*everyone loves Alex*".sem = ("*everyone*".sem, "*loves Alex*".sem)
$=(\lambda P.\forall x.(\text{PERSON}(x) \implies P(x))\ \lambda x.\text{LOVES}(x,\text{ALEX}))$
$=\forall x.(\text{PERSON}(x) \implies \text{LOVES}(x,\text{ALEX}))$

"*loves someone*".sem = ("*loves*".sem, "*someone*".sem)
$=(\lambda P.\lambda x.P(\lambda y.\text{LOVES}(x, y))\ \lambda P.\exists y.(\text{PERSON}(y) \wedge P(y)))$
$=\lambda x.(\lambda P.\exists y.(\text{PERSON}(y) \wedge P(y))\lambda y.\text{LOVES}(x, y))$
$=\lambda x.(\exists y.(\text{PERSON}(y)\wedge\text{LOVES}(x, y)))$

"*Alex loves someone*".sem = ("*Alex*".sem, "*loves someone*".sem)
$=(\lambda P.P(\text{ALEX})\ \lambda x.(\exists y.(\text{PERSON}(y)\wedge\text{LOVES}(x, y))))$
$=\exists y.(\text{PERSON}(y)\wedge\text{LOVES}(\text{ALEX}, y)$

(c) The semantic representation of the sentence *"everyone loves someone"*:
"*everyone loves someone*".sem = ("*everyone*".sem, "*loves someone*".sem)
$=(\lambda P.\forall x.(\text{PERSON}(x) \implies P(x))\ \lambda x.(\exists y.(\text{PERSON}(y)\wedge\text{LOVES}(x, y))))$
$=\forall x.(\text{PERSON}(x) \implies \exists y.(\text{PERSON}(y)\wedge\text{LOVES}(x, y)))$

Question 4: Floyd-Warshall WFSA

(a) The accept status of the sample strings is shown in Table 1.

(b) The modified Floyd Algorithm is shown in Algorithm 1, and the runtime result is shown in Figure 5 and Figure 6. **Note that we initialize the self-weight to be the minimum of the self-loop weight and zero**.

(c) The number of iterations is bounded by the number of verticies (i.e. $|V|$), which equals to 6 in this question. The necessary condition of termination is that there is no cycle with negative weight in the graph (otherwise the algorithm can always find shorter path by going around the cycle).

(d) The complexity of Floyd-Warshall algorithm with additional second matrix is:

- Time: $O(|V|^3)$ (three nested for-loop)
- Memory: $O(2|V|^2) = O(|V|^2)$ (two 2D matrices)

The maximum time complexity for backtracking the "lowest-weight" path is $O|V|)$ since the largest possible path contains at most $|V|$ vertices and retrieve each vertex cost constant time with the second matrix. To retrieve the shortest path between all pairs of vertices cost $O(|V|^3)$.

number	sample strings	accepted	weight
1	is this assignment educational	no	
2	not this assignment is educational	no	
3	this assignment not	yes	8
4	educational is this not	no	
5	not not not educational	yes	25
6	course assignment is this	yes	8
7	this course assignment not educational	yes	14
8	not educational is not educational	yes	12
9	this assignment is not educational	yes	9
10	is this assignment not educational	yes	15
11	not educational is this	yes	10
12	this course is not not educational	yes	21
13	this course is interesting	no	
14	is this course assignment not educational	yes	18
15	this course assignment is not educational	yes	12
16	course assignment is not educational	yes	10
17	this assignment course is educational	no	

Table 1: Some strings from $\mathcal{Y}_{\geq 2, \leq 6}$

Algorithm 1 Modified Floyd Algorithm

```
distance ← [][]
path ← [][]
for i = 1 to |V| do
    for j = 1 to |V| do
        if there is an edge i → j then
            distance[i][j] ← weight(i → j)
            path[i][j] ← j
        else
            distance[i][j] ← ∞
            path[i][j] ← None
        end if
    end for
    distance[i][i] ← min(0, distance[i][i])
end for
for k = 1 to |V| do
    for i = 1 to |V| do
        for j = 1 to |V| do
            if distance[i][j] > distance[i][k] + distance[k][j] then
                distance[i][j] ← distance[i][k] + distance[k][k] + distance[k][j])
                path[i][j] ← path[i][k]
            end if
        end for
    end for
end for
```

iteration: n = 0, weight matrix

	a this	b assign-ment	c course	d is	e not	f edu-cational
a this	0	1	2	∞	∞	∞
b assignment	∞	0	∞	1	5	∞
c course	∞	2	0	2	3	∞
d is	4	∞	∞	0	2	4
e not	∞	∞	∞	∞	0	2
f educational	∞	∞	∞	3	∞	0

iteration: n = 0, backtracking matrix

	a this	b assign-ment	c course	d is	e not	f edu-cational
a this	a	b	c	None	None	None
b assignment	None	b	None	d	e	None
c course	None	b	c	d	e	None
d is	a	None	None	d	e	f
e not	None	None	None	None	e	f
f educational	None	None	None	d	None	f

iteration: n = 1, weight matrix

	a this	b assign-ment	c course	d is	e not	f edu-cational
a this	0	1	2	∞	∞	∞
b assignment	∞	0	∞	1	5	∞
c course	∞	2	0	2	3	∞
d is	4	5	6	0	2	4
e not	∞	∞	∞	∞	0	2
f educational	∞	∞	∞	3	∞	0

iteration: n = 1, backtracking matrix

	a this	b assign-ment	c course	d is	e not	f edu-cational
a this	a	b	c	None	None	None
b assignment	None	b	None	d	e	None
c course	None	b	c	d	e	None
d is	a	a	a	d	e	f
e not	None	None	None	None	e	f
f educational	None	None	None	d	None	f

iteration: n = 2, weight matrix

	a this	b assign-ment	c course	d is	e not	f edu-cational
a this	0	1	2	2	6	∞
b assignment	∞	0	∞	1	5	∞
c course	∞	2	0	2	3	∞
d is	4	5	6	0	2	4
e not	∞	∞	∞	∞	0	2
f educational	∞	∞	∞	3	∞	0

iteration: n = 2, backtracking matrix

	a this	b assign-ment	c course	d is	e not	f edu-cational
a this	a	b	c	b	b	None
b assignment	None	b	None	d	e	None
c course	None	b	c	d	e	None
d is	a	a	a	d	e	f
e not	None	None	None	None	e	f
f educational	None	None	None	d	None	f

iteration: n = 3, weight matrix

	a this	b assign-ment	c course	d is	e not	f edu-cational
a this	0	1	2	2	5	∞
b assignment	∞	0	∞	1	5	∞
c course	∞	2	0	2	3	∞
d is	4	5	6	0	2	4
e not	∞	∞	∞	∞	0	2
f educational	∞	∞	∞	3	∞	0

iteration: n = 3, backtracking matrix

	a this	b assign-ment	c course	d is	e not	f edu-cational
a this	a	b	c	b	c	None
b assignment	None	b	None	d	e	None
c course	None	b	c	d	e	None
d is	a	a	a	d	e	f
e not	None	None	None	None	e	f
f educational	None	None	None	d	None	f

Figure 5: Floyd-Warshall algorithm, iteration 0 to 3; left column matrix should contain weights after iteration n; right column matrix should be iteratively filled for backtracking each path

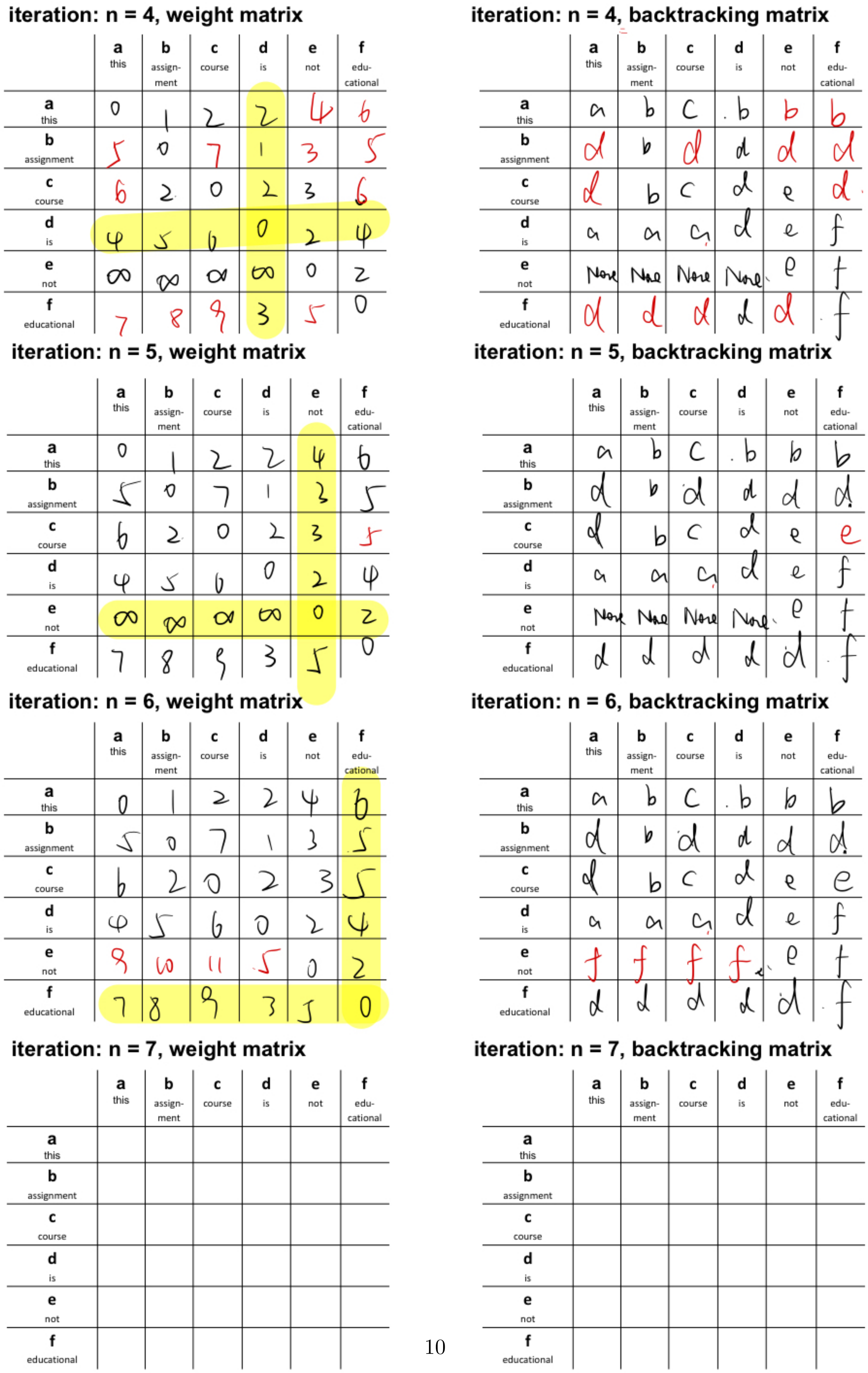

Figure 6: Floyd-Warshall algorithm, iteration 4 to 7; left column matrix should contain weights after iteration n; right column matrix should be iteratively filled for backtracking each path