COMS W4705: Homework 2 Written Problems

Steven Liu| xl2948@columbia.edu
(Used one late day)

2.1 Coding Reflections

In 1.2.3, I tried the following extensions:
- Changes to the preprocessing of the data (Tweet Tokenizer in NLTK), and
- Different strategies for optimization and training (adding a learning rate scheduler).

Extension 1:

For Tweet Tokenizer, I added a Boolean parameter, tweet_tokenizer, to the __init__ function of the class EmotionDataset (Line 22, utils.py), make_vectors() function (Line 74, utils.py), get_tokens() function (Line 96, utils.py), and vectorize_data() function (Line 132, utils.py). As a result, when hw2.py calls utils.vectorize_data() with tweet_tokenizer set to True (Line 238, hw2.py), the EmotionDataset objects (train_data, test_data and dev_data) will have their tweet_tokenizer parameters set to true as well (Line 161-163, utils.py). The EmotionDataset class will finally call get_tokens() with tweet_tokenizer = true (Line 42-43, utils.py), which will use nltk.TweetTokenizer() instead of nltk.word_tokenize to tokenize the data.

The main difference between the original tokenizer and Tweet Tokenizer is how they treat hashtags and other symbols, and since Twitter contains a lot of hashtags and symbols, I thought using Tweet Tokenizer would better tokenize the data and utilize features of Tweets.

The result given by running dense network using Tweet Tokenizer is similar to running the dense network using the original tokenizer. For instance, in my most recent trial, the original tokenizer gave a F1 score of 0.44457, nltk.word_tokenize gave a F1 score of 0.45154. The reason why there isn't an obvious performance is probably because how the tokenizer treats hashtags and the number of hashtags and other symbols are not very important or useful information for sentiment analysis.

Extension 2:

I wrote a new train_model() function (train_model_with_scheduler(), line 92-139, hw2.py) with a learning rate scheduler.

Since learning rate scheduler can adjust the learning rate through epochs, I thought a lower learning rate could help the model better fit to the data and give more accurate results.

With the learning rate scheduler, the F1 score of the dense network improves by about 0.02. For example, in my most recent trial, the F1 increased to over 0.464. This is probably because the original model converged too quickly to a suboptimal solution, while a smaller step size enabled by the learning rate scheduler helps find a better solution and thus gives a higher F1 score.

2.2 Backpropagation with RNNs

2.2.1 Forward Propagation

1.

a)

$$h_{t=1} = f(q_{t=1}) = f(W_x^T x_1 + W_h^T h_0 + b_h)$$

$$= f(\begin{bmatrix} 1 \\ 3 \end{bmatrix} (-1) + \begin{bmatrix} 1 & 2 \\ 3 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 1 \end{bmatrix} + \begin{bmatrix} 1 \\ 2 \end{bmatrix})$$

$$= f(\begin{bmatrix} -1 \\ -3 \end{bmatrix} + \begin{bmatrix} 4 \\ 7 \end{bmatrix} + \begin{bmatrix} 1 \\ 2 \end{bmatrix})$$

$$= f(\begin{bmatrix} 4 \\ 6 \end{bmatrix})$$

$$= \begin{bmatrix} \frac{1}{1+e^{-4}} \\ \frac{1}{1+e^{-6}} \end{bmatrix}$$

$$= \begin{bmatrix} 0.9820 \\ 0.9975 \end{bmatrix}$$

b)

$$\hat{y}_{t=1}$$

$$= f(p_{t=1})$$

$$= f(W_y^T h_{t=1} + b_y)$$

$$= f(\begin{bmatrix} 3 & 1 \end{bmatrix} \begin{bmatrix} 0.9820 \\ 0.9975 \end{bmatrix} + [1])$$

$$= f([0.9820 * 3 + 0.9975 * 1 + 1])$$

$$= f([4.9435])$$

$$= [0.9929]$$

c)

$h_{t=2}$

$= f(q_{t=2})$

$= f(W^T x_{t=2} + W_h^T h_{t=1} + b_h)$

$= f(\begin{bmatrix} 1 \\ 3 \end{bmatrix} 0 + \begin{bmatrix} 1 & 2 \\ 3 & 1 \end{bmatrix} \begin{bmatrix} 0.9820 \\ 0.9975 \end{bmatrix} + \begin{bmatrix} 1 \\ 2 \end{bmatrix})$

$= f(\begin{bmatrix} 0.9820 * 1 + 0.9975 * 2 + 1 \\ 0.9820 * 3 + 0.9975 * 1 + 2 \end{bmatrix})$

$= f(\begin{bmatrix} 3.977 \\ 5.9435 \end{bmatrix})$

$= (\begin{bmatrix} 0.9816 \\ 0.9974 \end{bmatrix})$

d)

$\hat{y}_{t=2}$

$= f(p_{t=2})$

$= f(W_y^T h_{t=2} + b_y)$

$= f(\begin{bmatrix} 3 & 1 \end{bmatrix} \begin{bmatrix} 0.9816 \\ 0.9974 \end{bmatrix} + [1])$

$= f([0.9816 * 3 + 0.9974 * 1 + 1])$

$= f([4.9422]$

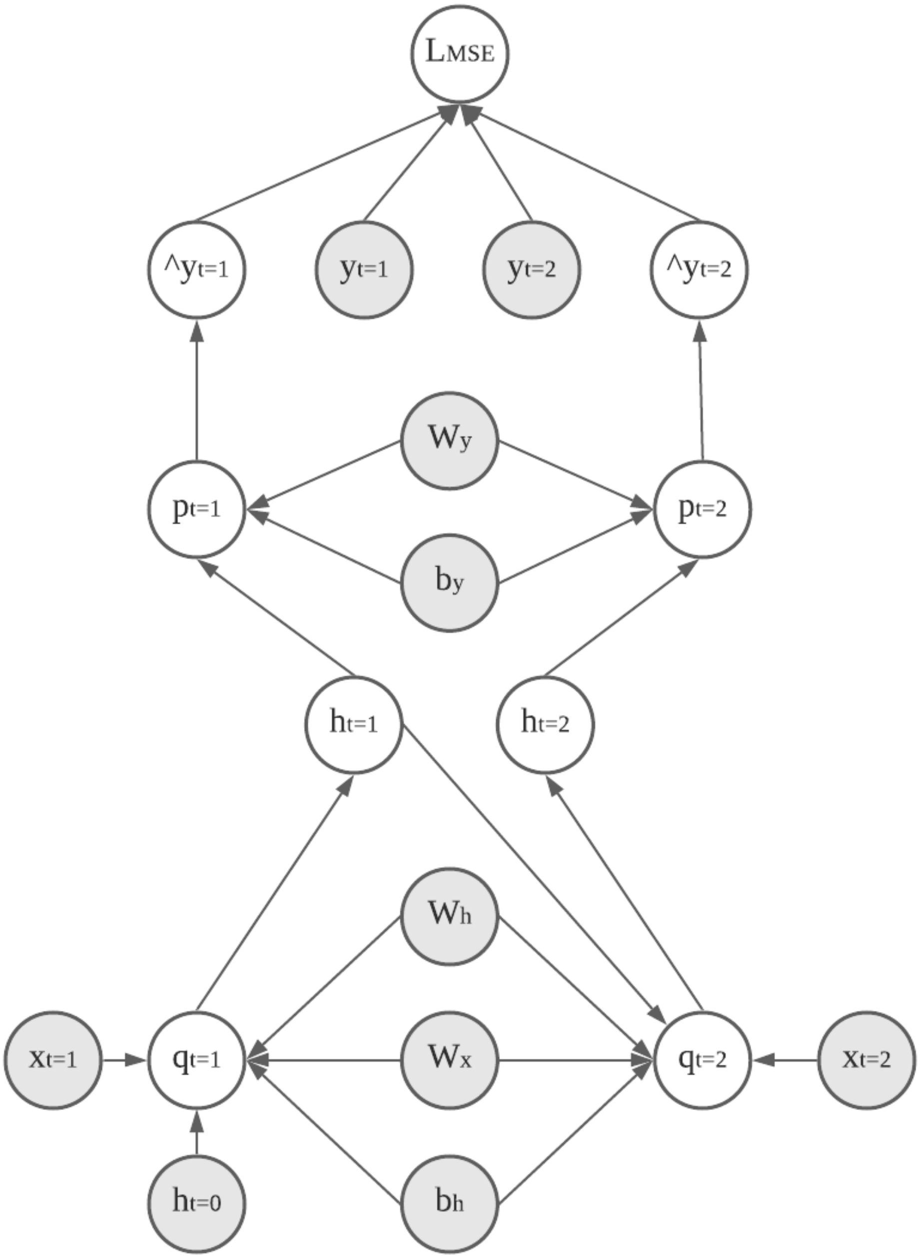
$= [0.9929]$

2.

$L_{MSE} = 1/1((\hat{y}_{t=1} - y_1)^2(\hat{y}_{t=2} - y_2)^2) = (0.9929 - 0)^2 + (0.9929 - 1)^2 = 0.9859$

2.2.2 Backpropagation Through Time

1. Computation Graph



2.Gradients

a)

$$\frac{\partial L_{MSE}}{\partial W_y}$$
$$= \frac{\partial L_{MSE}}{\partial \hat{y}_{t=1}} \frac{\partial \hat{y}_{t=1}}{\partial p_{t=1}} \frac{\partial p_{t=1}}{\partial W_y} + \frac{\partial L_{MSE}}{\partial \hat{y}_{t=2}} \frac{\partial \hat{y}_{t=2}}{\partial p_{t=2}} \frac{\partial p_{t=2}}{\partial W_y}$$
$$= \begin{bmatrix} 0.0136 \\ 0.0138 \end{bmatrix}$$

b)

$$\frac{\partial L_{MSE}}{\partial W_h}$$
$$= \frac{\partial L_{MSE}}{\partial \hat{y}_{t=1}} \frac{\partial \hat{y}_{t=1}}{\partial p_{t=1}} \frac{\partial p_{t=1}}{\partial h_{t=1}} \frac{\partial h_{t=1}}{\partial q_{t=1}} \frac{\partial q_{t=1}}{\partial W_h} + \frac{\partial L_{MSE}}{\partial \hat{y}_{t=2}} \frac{\partial \hat{y}_{t=2}}{\partial p_{t=2}} \frac{\partial p_{t=2}}{\partial h_{t=2}} \frac{\partial h_{t=2}}{\partial q_{t=2}} \frac{\partial q_{t=2}}{\partial W_h}$$
$$+ \frac{\partial L_{MSE}}{\partial \hat{y}_{t=2}} \frac{\partial \hat{y}_{t=2}}{\partial p_{t=2}} \frac{\partial p_{t=2}}{\partial h_{t=2}} \frac{\partial h_{t=2}}{\partial q_{t=2}} \frac{\partial q_{t=2}}{\partial h_{t=1}} \frac{\partial h_{t=1}}{\partial q_{t=1}} \frac{\partial q_{t=1}}{\partial W_h}$$
$$= \begin{bmatrix} 1.4736 * 10^{-3} \; 7.3408 * 10^{-4} \\ 6.8543 * 10^{-5} \; 3.4140 * 10^{-5} \end{bmatrix}$$

c)

$$\frac{\partial L_{MSE}}{\partial W_x}$$
$$= \frac{\partial L_{MSE}}{\partial \hat{y}_{t=1}} \frac{\partial \hat{y}_{t=1}}{\partial p_{t=1}} \frac{\partial p_{t=1}}{\partial h_{t=1}} \frac{\partial h_{t=1}}{\partial q_{t=1}} \frac{\partial q_{t=1}}{\partial W_x} + \frac{\partial L_{MSE}}{\partial \hat{y}_{t=2}} \frac{\partial \hat{y}_{t=2}}{\partial p_{t=2}} \frac{\partial p_{t=2}}{\partial h_{t=2}} \frac{\partial h_{t=2}}{\partial q_{t=2}} \frac{\partial q_{t=2}}{\partial W_x}$$
$$+ \frac{\partial L_{MSE}}{\partial \hat{y}_{t=2}} \frac{\partial \hat{y}_{t=2}}{\partial p_{t=2}} \frac{\partial p_{t=2}}{\partial h_{t=2}} \frac{\partial h_{t=2}}{\partial q_{t=2}} \frac{\partial q_{t=2}}{\partial h_{t=1}} \frac{\partial h_{t=1}}{\partial q_{t=1}} \frac{\partial q_{t=1}}{\partial W_x}$$
$$= [-7.3948 * 10^{-4} - 3.4399 * 10^{-5}]$$

d)

$$\frac{\partial L_{MSE}}{\partial b_y} =$$
$$= \frac{\partial L_{MSE}}{\partial \hat{y}_{t=1}} \frac{\partial \hat{y}_{t=1}}{\partial p_{t=1}} \frac{\partial p_{t=1}}{\partial b_y} + \frac{\partial L_{MSE}}{\partial \hat{y}_{t=2}} \frac{\partial \hat{y}_{t=2}}{\partial p_{t=2}} \frac{\partial p_{t=2}}{\partial b_y}$$
$$= [0.0139]$$

e)

$$\frac{\partial L_{MSE}}{\partial b_h} =$$
$$= \frac{\partial L_{MSE}}{\partial \hat{y}_{t=1}} \frac{\partial \hat{y}_{t=1}}{\partial p_{t=1}} \frac{\partial p_{t=1}}{\partial h_{t=1}} \frac{\partial h_{t=1}}{\partial q_{t=1}} \frac{\partial q_{t=1}}{\partial b_h} + \frac{\partial L_{MSE}}{\partial \hat{y}_{t=2}} \frac{\partial \hat{y}_{t=2}}{\partial p_{t=2}} \frac{\partial p_{t=2}}{\partial h_{t=2}} \frac{\partial h_{t=2}}{\partial q_{t=2}} \frac{\partial q_{t=2}}{\partial b_h}$$
$$+ \frac{\partial L_{MSE}}{\partial \hat{y}_{t=2}} \frac{\partial \hat{y}_{t=2}}{\partial p_{t=2}} \frac{\partial p_{t=2}}{\partial h_{t=2}} \frac{\partial h_{t=2}}{\partial q_{t=2}} \frac{\partial q_{t=2}}{\partial h_{t=1}} \frac{\partial h_{t=1}}{\partial q_{t=1}} \frac{\partial q_{t=1}}{\partial b_h}$$
$$= \begin{bmatrix} 7.3407 * 10^{-4} \\ 3.4139 * 10^{-5} \end{bmatrix}$$

3. Update Parameters

$$b_y = b_y - \eta \frac{\partial L_{MSE}}{\partial b_y}$$

$$= [1] - 0.01 * [0.0139]$$

$$= [0.999861]$$

$$W_h = W_h - \eta \frac{\partial L_{MSE}}{\partial W_h}$$

$$= \begin{bmatrix} 1 & 3 \\ 2 & 1 \end{bmatrix} - 0.01 * \begin{bmatrix} 1.4736 * 10^{-3} & 7.3408 * 10^{-4} \\ 6.8543 * 10^{-5} & 3.4140 * 10^{-5} \end{bmatrix}$$

$$= \begin{bmatrix} 0.999985264 & 2.9999926592 \\ 1.9999993146 & 0.9999996586 \end{bmatrix}$$

$$\approx \begin{bmatrix} 1.00 & 3.00 \\ 2.00 & 1.00 \end{bmatrix}$$

2.3 Grammar

1. CFG for parsing sentences

Analyzing the sentences, I got:

S → NP VP PUNC | NP VP SC PUNC
- (...)
- (The small boy / built a very large round snowman / who wore a red cap.)

SC (subordinate clause) → PRON VP
- (who / wore a red cap)

VP → V ADV PP | V PP | V NP | V NP NP
- (fell / quietly / in the city on Monday)
- (drifted / to 3 feet)
- (built / a very large snowman, wore / a red cup)
- (gave / the snowman / a carrot nose)

NP → DET N | DET N PP | DET AdjP N | NUM N | NP SC | DET N N
- (the / city, the / snow, the / park, the / snowman, the / boy)
- (The / snow / in the park)
- (The / soft white / snow, The / small / boy, a / red / cup, a / very large / snowman)
- (3 / feet)
- (a very large snowman / who wore a red cup)
- (a carrot nose)

AdjP → ADJ | ADV AdjP | ADJ AdjP
- (soft, white, small, large, round, red, large round)
- (very / large round)
- (soft / white, large / round)

PP → PP PP | ADP NP | ADP PROPN
- (in the city / on monday)
- (in / the city, in / the park, to / 3 feet)
- (on / monday)

ADJ → soft | white | large | round | red
ADV → very | quietly
ADP → in | on | to
DET → a | the
N → snow | city | Monday | park | feet | boy | snowman | cap | carrot | nose
NUM → 3
PRON → who
PROPN → Monday
PUNC → .
V → fell | drifted | built | wore | gave

Thus, I define the CFG, **G**, as this:

G = (**V, T, P, S**), where

S is the start symbol,

V = {S, SC, VP, NP, AdjP, PP},

T = {ADJ, ADV, ADP, DET, N, NUM, PRON, PROPN, PUNC, V},

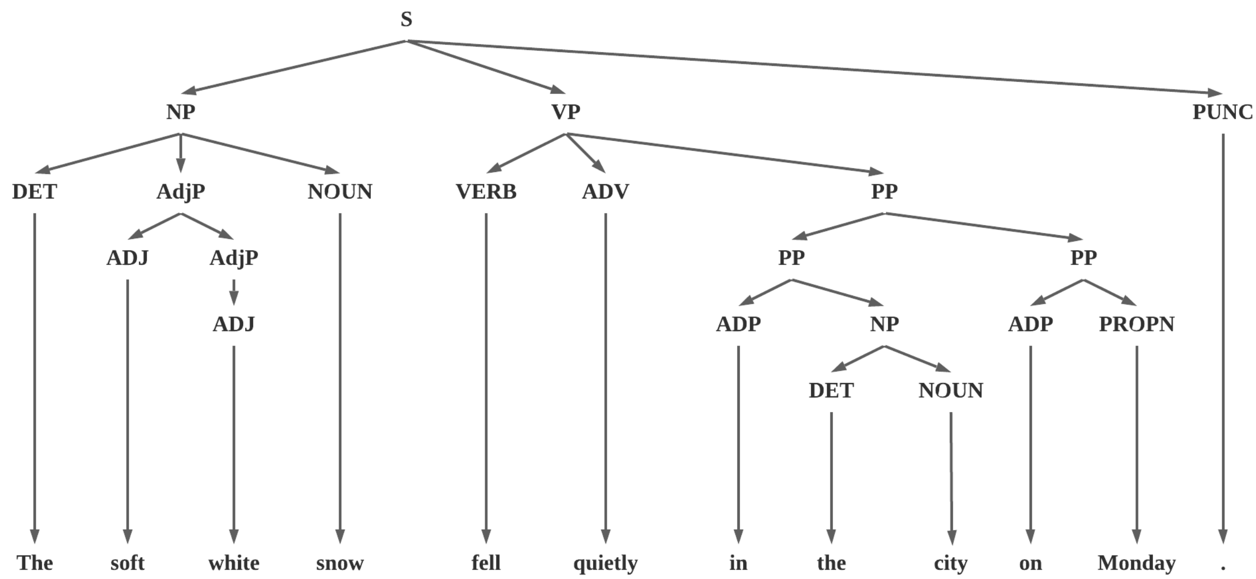
P:

S → NP VP PUNC | NP VP SC PUNC
SC → PRON VP
VP → VERB ADV PP | VERB PP | VERB NP | VERB NP NP
NP → DET NOUN | DET NOUN PP | DET AdjP NOUN | NUM NOUN | NP SC
| DET NOUN NOUN
AdjP → ADJ | ADV AdjP | ADJ AdjP
PP → PP PP | ADP NP | ADP PROPN
ADJ → soft | white | large | round | red
ADV → very | quietly
ADP → in | on | to
DET → a | the
NOUN → snow | city | Monday | park | feet | boy | snowman | cap | carrot | nose
NUM → 3
PRON → who
PROPN → Monday
PUNC → .
VERB → fell | drifted | built | wore | gave

As proven earlier, **G** can be used to parse sentences (a) through (d). It has 16 rules and will not allow ungrammatical sentences to be generated.

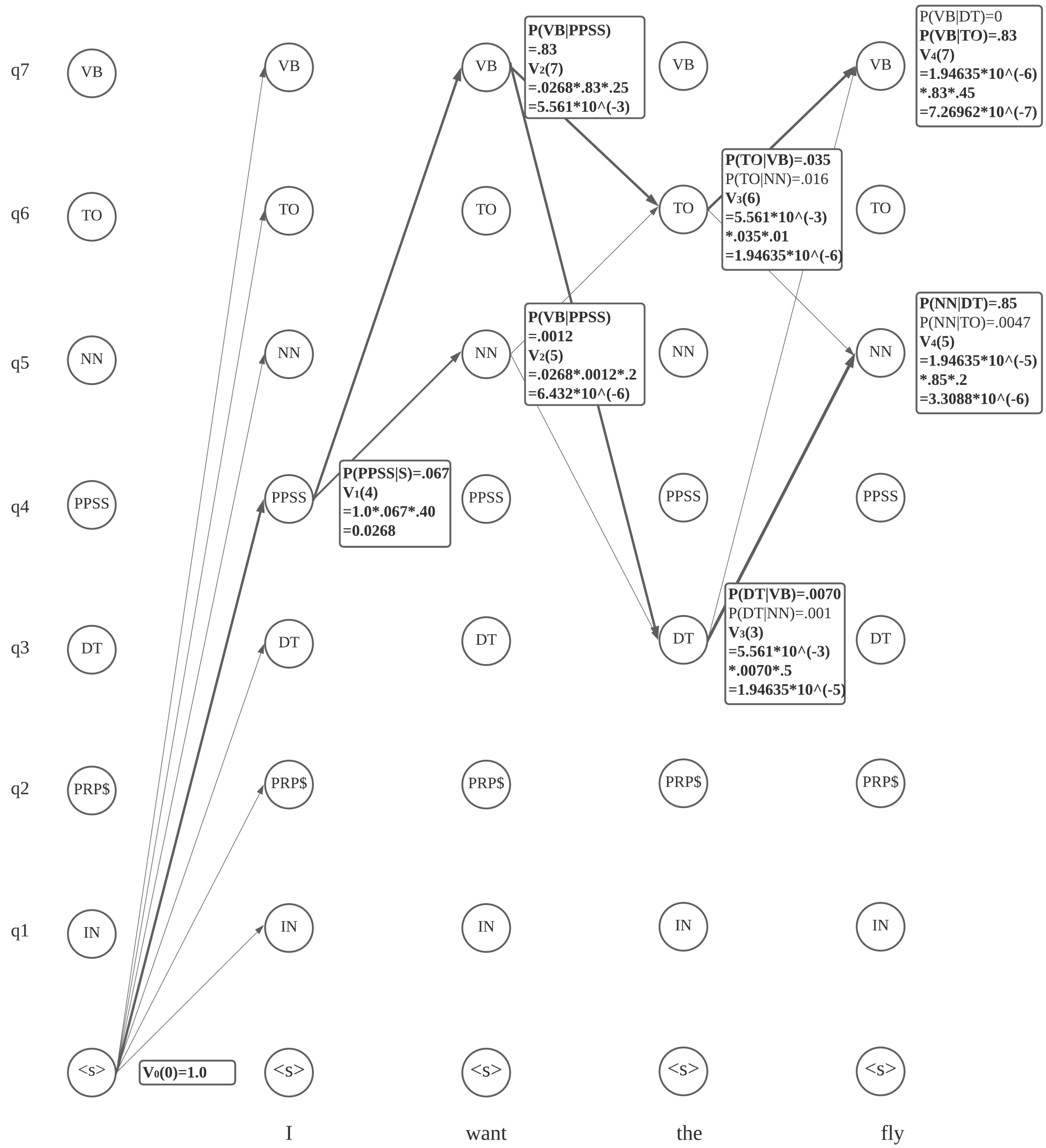
2. Parse Tree

There is only one possible parse tree that allows my grammar to generate sentence (a):



2.4 Viterbi Algorithm

1. Dynamic programming trellis



2. Formula

Formula with values to compute the probability of "fly" as either verb or noun:

$$P(verb)$$
$$= P(<s>) * P(PPSS| <s>) * P(PPSS| \text{" } I \text{ "}) * P(VN|PPSS) * P(VN| \text{" } want \text{ "}) * P(TO|VB)$$
$$* P(TO| \text{" } the \text{ "}) * P(VB|TO) * P(VB| \text{" } fly \text{ "})$$
$$= 1.0 * .067 * .4 * .83 * .25 * .035 * .01 * .83 * .45$$
$$= 7.26962 * 10^{-7}$$
$$P(noun)$$
$$= P(<s>) * P(PPSS| <s>) * P(PPSS| \text{" } I \text{ "}) * P(VN|PPSS) * P(VN| \text{" } want \text{ "}) * P(DT|VB)$$
$$* P(DT| \text{" } the \text{ "}) * P(NN|DT) * P(NN| \text{" } fly \text{ "})$$
$$= 1.0 * .067 * .4 * .83 * .25 * .0070 * .5 * .85 * .2$$
$$= 3.3088 * 10^{-6}$$