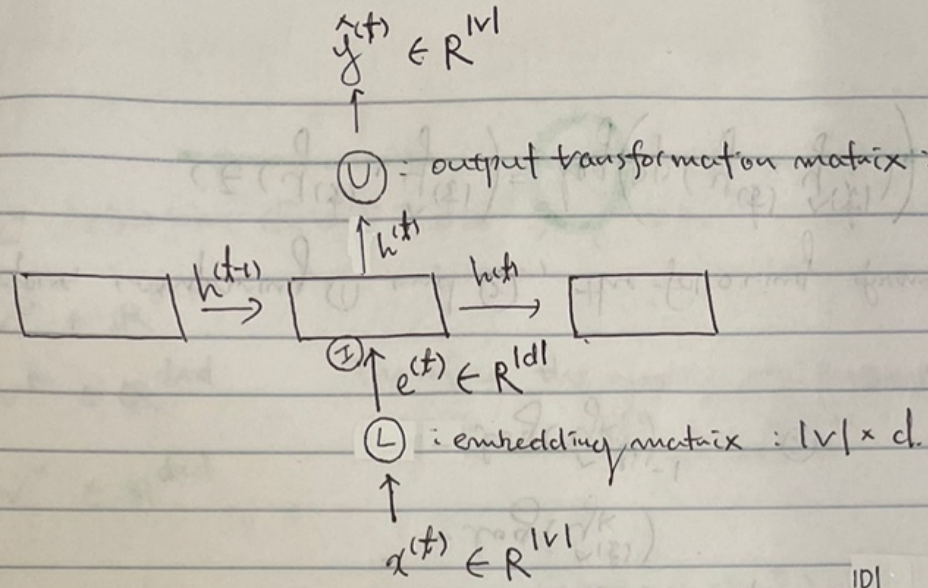
1. (a)



$$\underbrace{h^{(t)}}_{\in R^{|D|}} = \text{sigmoid}(\underbrace{h^{(t-1)}}_{\in R^{|D|}} \underbrace{H}_{\in R^{|D|\times|D|}} + \underbrace{e^{(t)}}_{\in R^{|d|}} \underbrace{I}_{\in R^{|d|\times|D|}} + \underbrace{b_1}_{\in R^{|D|}})$$

$$\hat{y}^{(t)} = \text{softmax}(\underbrace{h^{(t)}}_{\in R^{|D|}} \cdot \underbrace{U}_{\in R^{|D|\times|V|}} + \underbrace{b_2}_{\in R^{|V|}})$$

Summarizing the above,

$$\boxed{\begin{aligned} &L \in R^{|V|\times d} \\ &H \in R^{D\times D} \\ &I \in R^{d\times D} \\ &U \in R^{D\times|V|} \\ &b_1 \in 1\times D \\ &b_2 \in 1\times|V| \end{aligned}}$$

d: dimension of word embedding.

D: # of hidden units.

1-(b)

Suppose $y_k^{(t)}$ is the non-zero element in $y^{(t)}$. In visualization,

$$|V| \left\{ \begin{array}{c} \\ \\ \\ \\ \\ \\ \\ \end{array} \right. \underbrace{\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix}}_{y^{(t)}} \begin{array}{l} \text{0th} \\ \text{1st} \\ \text{2nd} \\ \vdots \\ \text{kth} \\ \vdots \\ (|V|-1)\text{th} \end{array} \qquad \underbrace{\begin{bmatrix} \vdots \\ \vdots \\ \star \\ \vdots \\ \vdots \end{bmatrix}}_{\hat{y}^{(t)}}$$

"Perplexity" is defined as the inverse probability of the target word according to the model prediction $\bar{P}$. This can be written as the following.

$$\left(\sum_{j=1}^{|V|} y_j^{(t)} \cdot \hat{y}_j^{(t)} \right)^{-1} = \left(\hat{y}_k^{(t)} \right)^{-1} \cdots\cdots ①$$

"Cross entropy" we use the following formula.

$$CE(y^{(t)}, \hat{y}^{(t)}) = -\sum_{j=1}^{|V|} \underbrace{y_j^{(t)}}_{\text{true}} \log(\underbrace{\hat{y}_j^{(t)}}_{\text{prediction.}})$$

$$= -\log(\hat{y}_k^{(t)})$$

$$= \log(\hat{y}_k^{(t)})^{-1} \cdots ②$$

Therefore, comparing ① and ②, the following formula holds.

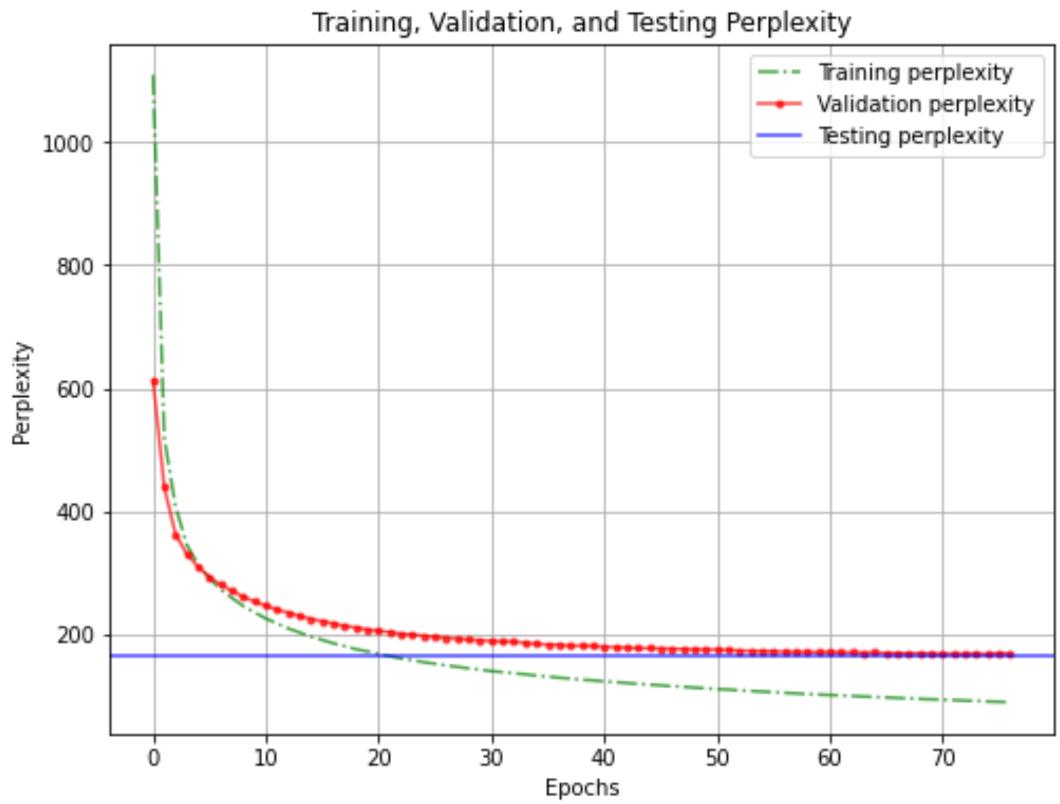
$$CE(y^{(t)}, \hat{y}^{(t)}) = \log(PP(y^{(t)}, \hat{y}^{(t)}))$$

Result and analysis of the programming part

Jin Hyun Park, UIN:633001823

1-c.

Training, validation, and testing perplexity:



Best hyperparameters:

```
- batch_size = 64
- embed_size = 256
- hidden_size = 2014
- num_steps = 10
- max_epochs = 100
- early_stopping = 2
- dropout = 0.1
- learning_rate = 0.001
- optimizer = SGD
```

Best testing perplexity score:

- 165.516

Sample of results:

- input: in palo alto
    - output: in palo alto to ms. also said it would be expected to begin N to N miles from N N N in N for fees in consumer without regulatory <unk> <eos>
- input: what is your name
    - output: what is your name earlier and trouble <eos>
- input: what is your goal
    - output: what is your goal last teachers put through and use of the u.s. criteria or settled else <eos>
- input: i like you
    - output: i like you could imagine <unk> of panic a sigh on saving sales and <unk> highways which <eos>
- input: stock market today
    - output: stock market today because <eos>
- input: what time is it
    - output: what time is it is and plaintiffs ' plans a big stocks in the abortions of campbell soup co. fifth four <unk> aircraft and merrill 's meeting employees in earth <eos>

It was able to see that the sentence that the model generated does not really make sense. This implies that RNN itself has a limitation in generating texts, leading to the development of sophisticated models such as Transformers. It might be possible to get a better result by tuning hyperparameters.

2.(a)

single-head: $\underset{\sim}{A_S} = \text{softmax}(QW^Q(KW^K)^T) \cdot VW^V$

$10\times512 \quad 512\times10 \quad 10\times512 \Rightarrow 10\times512$

$W^Q \in \mathbb{R}^{d\times d}$

$W^K \in \mathbb{R}^{d\times d}$ Therefore, the number of parameters for the single-head attention is $d^2 \times 3 = 3d^2$.

$W^V \in \mathbb{R}^{d\times d}$

multi-head: $\underset{\sim}{A_M} = (\text{concat}(\text{softmax}(QW_i^Q(KW_i^K)^T) \cdot VW_i^V))\, W^O$

$10\times64 \quad 64\times10 \quad 10\times64 \Rightarrow 10\times64$

$\Rightarrow 10\times512$

$W^Q \in \mathbb{R}^{d\times d/h}$

$W^K \in \mathbb{R}^{d\times d/h}$

$W^V \in \mathbb{R}^{d\times d/v}$

$W^O \in \mathbb{R}^{d\times d}$

$$\underbrace{\underbrace{d\times d/h \times 3}_{\text{single-head attention}} \times h}_{\text{Multi-head attention (concatenation)}} + \underbrace{d^2}_{\text{additional matrix } (W^O)} = 4d^2$$

if we don't think about W_O then the answer is $3d^2$

2.(b)

<Single head attention>

For generating $\tilde{Q} = Q \cdot W^Q$, $n\times d \quad d\times d$: $O(nd^2)$

" $\tilde{K} = K \cdot W^K$, $n\times d \quad d\times d$: $O(nd^2)$

$\tilde{V} = V \cdot W^V$, $n\times d \quad d\times d$: $O(nd^2)$

$\text{Softmax}(\tilde{Q} \cdot \tilde{K}^T) =$ $n\times d \quad d\times n$: $O(n^2 d + \underbrace{nd}_{\text{transpose}} + \underbrace{n^2}_{\text{softmax}})$

$\text{Softmax}(\underset{n\times n}{\tilde{Q}\cdot\tilde{K}^T}) \cdot \underset{n\cdot d}{\tilde{V}} \Rightarrow O(n^2 d)$

Summing up all of the O notation,

$$O(3nd^2 + 2n^2 d + nd + n^2)$$

$$\rightarrow O(nd^2 + n^2 d)$$

<Multi-head attention>

For generating $\tilde{Q} = Q \cdot W^Q = n \times d \cdot d \times d/h = O(nd \cdot \frac{d}{h})$

$$\tilde{K} = K \cdot W^K = n \times d \cdot d \times d/h = O(n \cdot d \cdot \frac{d}{h})$$

$$\tilde{V} = V \cdot W^V = n \times d \quad d \times d/h = O(n \cdot d \cdot \frac{d}{h})$$

$$\text{Softmax}(\tilde{Q} \cdot \tilde{K}^T) = n \cdot d/h \cdot d/h \cdot n = O(n^2 \frac{d}{h} + \frac{d}{h} \cdot n + n^2)$$

$$\underbrace{\text{Softmax}(\tilde{Q} \cdot \tilde{K}^T)}_{n \times n} \cdot \underbrace{\tilde{V}}_{n \cdot d/h} \Rightarrow O(n^2 \cdot d/h)$$

Summing up all O notations,

$$O\left(3\frac{nd^2}{h} + 2n^2 d/h + \frac{nd}{h} + n^2\right)$$

We need to do this h times,

$$O\left(h\left(\frac{3nd^2}{h} + \frac{2n^2 d}{h} + \frac{nd}{h} + n^2\right)\right)$$

$$= O(3nd^2 + 2n^2 d + nd + hn^2)$$

$$= O(3nd^2 + 2n^2 d)$$

hn^2, this negligible because it is a constant.

$$= O(nd^2 + n^2 d) \cdots ①$$

Considering W^O, we need to add $n \times d \cdot d \cdot d \Rightarrow nd^2$ to ①.
But, the time complexity remains the same.

Therefore, $O(nd^2 + n^2 d)$

We can verify that single head's and multihead attention's time complexity is similar to each other.

7.(a).

Adjacency matrix A is $n \times n$ matrix where n is # of nodes.
We need to modify A such that A also contains its node itself. (i.e. all nodes have a self-loop)

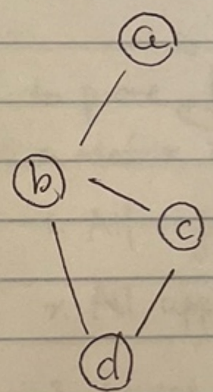
$$\tilde{A} \leftarrow \boxed{A + I_n}$$

Will add self-loop to each node.

7(b).

Suppose we have the following graph G.

Then the adjacency matrix $\tilde{A}$ is



<Graph G>

$$\tilde{A} = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{pmatrix}$$

To normalise $\tilde{A}$ we should first build a matrix D.

$$\tilde{D} = \begin{pmatrix} 2 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 3 \end{pmatrix}$$

where D is a diagonal matrix and each component of diagonal element is Σ of row of $\tilde{A}$

Let $\hat{A}$ is a normalised matrix the we can write

$$\hat{A} = \tilde{D}^{-1/2} \tilde{A} \tilde{D}^{-1/2}$$

and using $\hat{A}$ instead of $\tilde{A}$ (or A) ensures the scale of feature vectors to be maintained.