

Assignment 2

a) $-\sum_{w \in V} y_w \log(\hat{y}_w) = -y_1 \log(\hat{y}_1) - y_2 \log(\hat{y}_2) - y_3 \log(\hat{y}_3) - \dots - y_n \log(\hat{y}_n) = -y_0 \log(\hat{y}_0) = -\log(\hat{y}_0)$

b)

$$P(O=o | C=c) = \frac{\exp(u_c^\top v_c)}{\sum_{w \in V} \exp(u_w^\top v_c)}$$

$$J(v_c, o, U) = -\log P(O=o | C=c) = -\log \left(\frac{\exp(u_o^\top v_c)}{\sum_{w \in V} \exp(u_w^\top v_c)} \right) = -\log(\exp(u_o^\top v_c)) + \log(\sum_{w \in V} \exp(u_w^\top v_c))$$

$$\begin{aligned} \frac{\partial}{\partial v_c} J(v_c, o, U) &= -\frac{\exp(u_o^\top v_c) u_o}{\exp(u_o^\top v_c)} + \frac{1}{\sum_{w \in V} \exp(u_w^\top v_c)} \frac{\partial}{\partial v_c} \sum_{z \in V} \exp(u_z^\top v_c) \\ &= -u_o + \frac{1}{\sum_{w \in V} \exp(u_w^\top v_c)} \sum_{z \in V} \exp(u_z^\top v_c) u_z \\ &= -u_o + \sum_{z \in V} \frac{\exp(u_z^\top v_c)}{\sum_{w \in V} \exp(u_w^\top v_c)} u_z \\ &= -u_o + \sum_{z \in V} \hat{y}_z u_z \end{aligned}$$

c) Case 1: $w \neq o$

$$\begin{aligned} \frac{\partial}{\partial u_w} J(v_c, o, U) &= \frac{\partial}{\partial u_w} \log \left(\sum_{x \in V} \exp(u_x^\top v_c) \right) \\ &= \frac{1}{\sum_{x \in V} \exp(u_x^\top v_c)} \exp(u_w^\top v_c) v_c \\ &= v_c \hat{y}_w \end{aligned}$$

Case 2: $w = o$

$$\begin{aligned} \frac{\partial}{\partial u_o} J(v_c, o, U) &= -\frac{\exp(u_o^\top v_c)}{\exp(u_o^\top v_c)} v_c + \frac{1}{\sum_{x \in V} \exp(u_x^\top v_c)} \exp(u_o^\top v_c) v_c \\ &= -v_c + \hat{y}_o v_c \\ &= v_c (\hat{y}_o - 1) \end{aligned}$$

d)

$$\frac{\partial}{\partial u_x} J(v_c, o, U) = \begin{cases} v_c (\hat{y}_w - 1) & w = o \\ v_c \hat{y}_w & w \neq o \end{cases}$$

e)

$$\sigma(x) = \frac{e^x}{e^x + 1} = e^x (e^x + 1)^{-1}$$

$$\begin{aligned}\frac{\partial \sigma(x)}{\partial x} &= e^x (e^x + 1)^{-1} - e^x (e^x + 1)^{-2} e^x \\ &= \sigma(x) - \frac{e^{2x}}{(e^x + 1)^2} = \sigma(x) - \sigma^2(x) \\ &= \sigma(x)(1 - \sigma(x))\end{aligned}$$

f)

$$J_{\text{neg-example}}(v_c, o, U) = -\log(\sigma(u_o^\top v_c)) - \sum_{k=1}^K \log(\sigma(-u_k^\top v_c))$$

$$\begin{aligned}\frac{\partial}{\partial U_k} J(v_c, o, U) &= -\frac{1}{\sigma(u_o^\top v_c)} \sigma(u_o^\top v_c)(1 - \sigma(u_o^\top v_c)) u_o + \sum_{k=1}^K -\frac{1}{\sigma(-u_k^\top v_c)} \sigma(-u_k^\top v_c)(1 - \sigma(-u_k^\top v_c)) u_k \\ &= -(1 - \sigma(u_o^\top v_c)) u_o + \sum_{k=1}^K (1 - \sigma(-u_k^\top v_c)) u_k\end{aligned}$$

$$\frac{\partial}{\partial U_0} J(v_c, o, U) = -(1 - \sigma(u_o^\top v_c)) u_c$$

$$\frac{\partial}{\partial U_k} J(v_c, o, U) = (1 - \sigma(-u_k^\top v_c)) u_c$$

This loss function is more efficient because we don't have to compute the denominator of the softmax over all words in the vocabulary.

$$\sum_{x \in V} \exp(u_x^\top v_c)$$

g)

$$\begin{aligned}J_{\text{neg-example}}(v_c, o, U) &= -\log(\sigma(u_o^\top v_c)) - \sum_{w=k}^K \log(\sigma(-u_w^\top v_c)) - \sum_{w \neq k}^K \log(\sigma(-u_w^\top v_c)) \\ \frac{\partial}{\partial U_k} J_{\text{neg-example}}(v_c, o, U) &= \sum_{w \neq k}^K (1 - \sigma(-u_w^\top v_c)) v_c\end{aligned}$$

h)

$$i) \quad \frac{\partial}{\partial U} J_{\text{dip-gmm}}(v_c, w_{t-m}, \dots, w_{t+m}, U) = \sum_{\substack{-m \leq j \leq m \\ j \neq 0}} \frac{\partial}{\partial U} J(v_c, w_{t+j}, U)$$

$$ii) \quad \frac{\partial}{\partial U_c} J_{\text{dip-gmm}}(v_c, w_{t-m}, \dots, w_{t+m}, U) = \sum_{\substack{-m \leq j \leq m \\ j \neq 0}} \frac{\partial}{\partial U_c} J(v_c, w_{t+j}, U)$$

$$iii) \quad \frac{\partial}{\partial U_w} J_{\text{dip-gmm}}(v_c, w_{t-m}, \dots, w_{t+m}, U) = \sum_{\substack{-m \leq j \leq m \\ j \neq 0}} \frac{\partial}{\partial U_w} J(v_c, w_{t+j}, U), \quad w \neq c$$

$$= 0$$

Assignment 3

- 1a) i) By first building up momentum before making gradient updates you can accumulate otherwise small gradients and accelerate gradient updates in directions with small but consistent slopes. At the same time you prevent updates at points with large gradients from overshooting minima. It's like a ball rolling down a valley.
- ii) Weights which frequently receive large updates will have their updates scaled down. This can again help with the overshooting problem of gradient descent.
 Weights which haven't received large updates recently will get larger updates.
- b) i) $\beta = \frac{1}{P_{\text{train}}}$ so that the average size of the layer activations is the same during training and testing. Otherwise the next layer would see $\frac{1}{P_{\text{test}}}$ larger activations during testing which will probably result in poor accuracy.
- ii) Dropout acts as a kind of regularization during training and creates a kind of ensemble of subnetworks in the network which can again reduce generalization error. You shouldn't use it during testing because you'd basically be losing most effects of it as well as a (huge) part of the trained weights.

2a)

Stack	Buffer	New dependency	Transition
ROOT] , parsed, this, sentence, correctly		Initial
ROOT,]	parsed, this, sentence, correctly		Shift
ROOT,] , parsed	this, sentence, correctly	parsed \rightarrow]	Left-arc
ROOT, parsed, this	sentence, correctly		Shift
ROOT, parsed, this, sentence	correctly		Shift
ROOT, parsed, sentence	correctly	sentence \rightarrow this	Left-arc
ROOT, parsed	correctly	parsed \rightarrow sentence	Right-arc

ROOT, parsed correctly

ROOT, parsed

ROOT

Shift

parsed → correctly Right-Arc

ROOT → parsed Right-Arc

b) A sentence with n words will be parsed in $2n$ steps because for each word you first have to shift it onto the stack and then parse it.

f) i) Error type: Verb phrase attachment error

Incorrect dependency: wedding → fearing

Correct dependency: heading → fearing

ii) Error type: Coordination attachment error

Incorrect dependency: makes → rescue

Correct dependency: rush → rescue

iii) Error type: Prepositional phrase attachment error

Incorrect dependency: named → Midland

Correct dependency: guy → Midland

iv) Error type: Modifier attachment error

Incorrect dependency: elements → most

Correct dependency: crucial → most

Assignment 4

1)

g) The softmax for the masked input is 0, this prevents the decoder from attending to the <pad> tokens.

i) i) Dot product attention is faster to compute.

Multiplicative attention can learn a richer type of attention.

ii) Additive attention can again learn a richer kind of attention by learning two weight matrices and adding the result of their multiplication with the two vectors.

It's again more expensive to compute and more parameters to store / learn.

2)

a) Polysynthetic languages form sentences by concatenating subwords into words and thus it would be infeasible to create an embedding for each word ("ankue").

b) There are much fewer characters or subwords than there are words, as those are made from characters or subwords. Thus there isn't as much meaning to be encoded in their respective embeddings.

c) The intuition behind multilingual MT is that the learning signal from one language should also benefit the quality of translation of other languages. This is a kind of transfer learning.

d) i) One possible reason for this error might be that the word order differs between the Cherokee and English sentences.

Maybe try more LSTM layers or a richer kind of attention.

ii) Ulukeludi has multiple meanings and it might be difficult to pick the correct one.

Maybe more data or different attention

iii) *hüttefisch* is a very uncommon word and while the semantic meaning of the translation is correct, it is very difficult for the model to learn names.
Maybe replace with `<unk>` token.

- e) i) The model has learned to memorize phrases from the training data.
ii) The model's decoding reward isn't able to learn long term dependencies when decoding a translation.

f) i)

$$c_1) p_1 = \frac{0+1+1+1+0}{5} = \frac{3}{5}$$

$$p_2 = \frac{0+1+1+0}{4} = \frac{1}{2}$$

$$BP = 1 \quad \text{len}(c_1) = 5 \quad \text{len}(r_1) = 5$$

$$\text{BLEU} = 1 \cdot \exp\left(\log\left(\frac{1}{2}\frac{3}{5}\right) + \log\left(\frac{1}{2}\frac{1}{2}\right)\right) = \exp\left(\log\left(\frac{1}{2}\frac{1}{20}\right)\right) = \frac{11}{20}$$

$$c_2) p_1 = \frac{1+1+0+1+1}{5} = \frac{4}{5}$$

$$p_2 = \frac{1+0+0+1}{4} = \frac{1}{2}$$

$$BP = 1 \quad \text{len}(c_2) = 5 \quad \text{len}(r_2) = 5$$

$$\text{BLEU} = 1 \cdot \exp\left(\log\left(\frac{1}{2}\frac{4}{5}\right) + \log\left(\frac{1}{2}\frac{1}{2}\right)\right) = \frac{13}{20}$$

According to BLEU, c_2 is the better translation, which I agree on.

ii)

$$c_1) p_1 = \frac{3}{5} \quad p_2 = \frac{2}{4}$$

$$\text{BLEU} = \frac{11}{20}$$

$$c_2) p_1 = \frac{2}{5} \quad p_2 = \frac{1}{4}$$

$$\text{BLEU} = \exp\left(\log\left(\frac{1}{2}\frac{2}{5}\right) + \log\left(\frac{1}{2}\frac{1}{4}\right)\right) = \frac{1}{40}$$

Now c_1 has a better BLEU score even though c_2 is still a better translation.

- iii) to seen in ii), with a limited amount of reference sentences worse translations can receive higher BLEU scores and decrease the performance of the system.
- iv)
 - + Automatic
 - Shallow assessment of translation quality
 - + not subjective
 - Informative value decreases with limited amount of reference sentences