CS 224N: Assignment 1

Peter Li   Wednesday 24th January, 2018

Problem 1: Softmax (10 pts)

1.1 (a) Softmax Invariance to Constant (5 pts)

Answer:

$$\forall i,$$

$$\begin{aligned}(softmax(x+c))_i &= \frac{e^{(x_i+c)}}{\sum_j e^{(x_j+c)}} = \frac{e^{x_i} \times e^c}{e^c \times \sum_j e^{x_j}} \\ &= \frac{e^{x_i}}{\sum_j e^{x_j}} = softmax(x)_i\end{aligned}$$

so,

$$softmax(x) = softmax(x+c)$$

1.2 (b) Softmax Coding (5 pts)

Answer:

See code: ~/`q1_softmax.py`.

Problem 2: Neural Network Basics (30 pts)

2.1 (a) Sigmoid Gradient (3 pts)

Answer:

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

2.2 (b) Softmax Gradient w/ Cross Entropy Loss (3 pts)

Answer:

$$CE(\mathbf{y}, \hat{\mathbf{y}}) = -\sum_i y_i log(\hat{y}_i)$$

As $\mathbf{y}$ *is one-hot vector, assume* $y_k = 1$*, then other* $y_i = 0, \forall i \neq k$

$$CE(\mathbf{y}, \hat{\mathbf{y}}) = -y_k log(\hat{y}_k) = -log(\hat{y}_k)$$

As, $\hat{y}_k = softmax(\boldsymbol{\theta}) = \frac{e^{\theta_k}}{\sum_j e^{\theta_j}}$

$$\begin{aligned} CE(\boldsymbol{\theta}) &= -log(\frac{e^{\theta_k}}{\sum_j e^{\theta_j}}) \\ &= -\theta_k + log(\sum_j e^{\theta j}) \end{aligned}$$

So, for $y_k = 1$

$$\begin{aligned} \frac{\partial CE(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} &= \frac{\partial}{\partial \boldsymbol{\theta}}(-\theta_k + log(\sum_j e^{\theta_j}) \\ &= -\frac{\partial \theta_k}{\partial \boldsymbol{\theta}} + \frac{\partial}{\partial \boldsymbol{\theta}} log(\sum_j e^{\theta_j}) \end{aligned} \tag{2.1}$$

And,

$$\frac{\partial \theta_k}{\partial \theta_i} = \begin{cases} 1 & \text{, if } i = k \\ 0 & \text{, if } i \neq k \end{cases}$$

So,

$$\frac{\partial \theta_k}{\partial \boldsymbol{\theta}} = \mathbf{y} \tag{2.2}$$

And,

$$\begin{aligned} \frac{\partial}{\partial \theta_i} log(\sum_j e^{\theta_j}) &= \frac{1}{\sum_j e^{\theta_j}} \frac{\partial}{\partial \theta_i}(\sum_j e^{\theta_j}) \\ &= \frac{1}{\sum_j e^{\theta_j}} \frac{\partial}{\partial \theta_i} e^{\theta_i} \\ &= \frac{e^{\theta_i}}{\sum_j e^{\theta_j}} \\ &= softmax(\theta_i) \end{aligned} \tag{2.3}$$

With above equations,

$$\frac{\partial CE(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = -\mathbf{y} + \hat{\mathbf{y}}$$

2.3 (c) One Hidden Layer Gradient (6 pts)

Answer:

To caculate $\frac{\partial J}{\partial x}$*, where* $J = CE(\mathbf{y}, \hat{\mathbf{y}})$

Let $\boldsymbol{\theta} = \boldsymbol{h}\boldsymbol{W}_2 + \boldsymbol{b}_2$, and $\boldsymbol{Z1} = \boldsymbol{x}\boldsymbol{W}_1 + \boldsymbol{b}_1$

and from 2(b), we know $\frac{\partial}{\partial \boldsymbol{\theta}} CE(\mathbf{y}, \hat{\mathbf{y}}) = -\boldsymbol{y} + \hat{\boldsymbol{y}}$

we get,

$$\begin{aligned}
\frac{\partial J}{\partial \boldsymbol{\theta}} &= \boldsymbol{\delta}_3 = \hat{\boldsymbol{y}} - \boldsymbol{y} \\
\frac{\partial J}{\partial \boldsymbol{h}} &= \boldsymbol{\delta}_2 = \boldsymbol{\delta}_3 \frac{\partial \boldsymbol{\theta}}{\partial \boldsymbol{h}} = \boldsymbol{\delta}_3 \boldsymbol{W}_2^T \\
\frac{\partial J}{\partial \boldsymbol{Z1}} &= \boldsymbol{\delta}_1 = \frac{\partial J}{\partial \boldsymbol{h}} \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{Z1}} = \boldsymbol{\delta}_2 \circ \sigma'(\boldsymbol{Z1}) \\
\frac{\partial J}{\partial \boldsymbol{x}} &= \boldsymbol{\delta}_1 \frac{\partial \boldsymbol{Z1}}{\partial \boldsymbol{x}} \\
&= \boldsymbol{\delta}_1 \boldsymbol{W}_1^T
\end{aligned}$$

2.4 (d) No. Parameters (2 pts)

Answer:

$\boldsymbol{W_1}$*,*$\boldsymbol{W_2}$*,*b_1, b_2 *are parameters. And* $\boldsymbol{W1} \in \mathbb{R}^{D_x \times H}$*,* $\boldsymbol{W2} \in \mathbb{R}^{H \times D_y}$*,* $b_1 \in \mathbb{R}^{H}$*,* $b_2 \in \mathbb{R}^{D_y}$

In total, parameter count is

$$(D_x \times H) + H + (H \times D_y) + D_y$$

2.5 (e) Sigmoid Activation Code (4 pts)

See code: ~/`q2_sigmoid.py`.

2.6 (f) Gradient Check Code (4 pts)

See code: ~/`q2_gradcheck.py`.

2.7 (g) Neural Net Code (8 pts)

See code: ~/`q2_neural.py`.

Problem 3: Word2Vec (40 pts + 2 bonus)

3.1 (a) Context Word Gradients (3 pts)

Answer:

set $z_i = \boldsymbol{u}_i^T \boldsymbol{v}_c$, *so* $\boldsymbol{Z} = \boldsymbol{U}^T \boldsymbol{v}_c$

From Problem 2.2 $\frac{\partial J}{\partial \boldsymbol{\theta}} = (\hat{\boldsymbol{y}} - \boldsymbol{y})$. Here, let $\boldsymbol{\theta} = \boldsymbol{Z}$.

And also note that $\boldsymbol{Z} \in \mathbb{R}^{V \times 1}$, $\boldsymbol{v}_c \in \mathbb{R}^{D \times 1}$, $\boldsymbol{U} \in \mathbb{R}^{D \times V}$, $\hat{\boldsymbol{y}} - \boldsymbol{y} \in \mathbb{R}^{V \times 1}$

$$\begin{aligned}\frac{\partial J}{\partial \boldsymbol{v}_c} &= \frac{\partial J}{\partial \boldsymbol{Z}} \frac{\partial \boldsymbol{Z}}{\partial \boldsymbol{v}_c} \\ &= \boldsymbol{U}(\hat{\boldsymbol{y}} - \boldsymbol{y})\end{aligned}$$

3.2 (b) Output Word Gradients (3 pts)

Answer:

Note that $\boldsymbol{Z} \in \mathbb{R}^{V \times 1}$, $\boldsymbol{v}_c \in \mathbb{R}^{D \times 1}$, $\boldsymbol{U} \in \mathbb{R}^{D \times V}$, $\hat{\boldsymbol{y}} - \boldsymbol{y} \in \mathbb{R}^{V \times 1}$

$$\begin{aligned}\frac{\partial J}{\partial \boldsymbol{U}} &= \frac{\partial J}{\partial \boldsymbol{Z}} \frac{\partial \boldsymbol{Z}}{\partial \boldsymbol{U}} \\ &= \boldsymbol{v}_c(\hat{\boldsymbol{y}} - \boldsymbol{y})^T\end{aligned}$$

3.3 (c) Repeat Gradients with Negative Sampling Loss (6 pts)

Answer:

$$\begin{aligned}
\frac{\partial J}{\partial \boldsymbol{v}_c} &= \frac{\partial}{\partial \boldsymbol{v}_c}((-log(\sigma(\boldsymbol{u}_o^T\boldsymbol{v}_c))) - \sum_{k=1}^{K} log(\sigma(-\boldsymbol{u}_k^T\boldsymbol{v}_c))) \\
&= -\frac{\partial}{\partial \boldsymbol{v}_c}(log(\sigma(\boldsymbol{u}_o^T\boldsymbol{v}_c))) - \sum_{k=1}^{K}\frac{\partial}{\partial \boldsymbol{v}_c} log(\sigma(-\boldsymbol{u}_k^T\boldsymbol{v}_c)) \\
&= \frac{-1}{\sigma(\boldsymbol{u}_o^T\boldsymbol{v}_c)}(\frac{\partial}{\partial \boldsymbol{v}_c}\sigma(\boldsymbol{u}_o^T\boldsymbol{v}_c)) - \sum_{k=1}^{K}\frac{1}{\sigma(-\boldsymbol{u}_k^T\boldsymbol{v}_c)}\frac{\partial}{\partial \boldsymbol{v}_c}\sigma(-\boldsymbol{u}_k^T\boldsymbol{v}_c) \\
&= -(1-\sigma(\boldsymbol{u}_o^T\boldsymbol{v}_c))\boldsymbol{u}_o - \sum_{k=1}^{K}(1-\sigma(-\boldsymbol{u}_k^T\boldsymbol{v}_c))(-\boldsymbol{u}_k) \\
&= (\sigma(\boldsymbol{u}_o^T\boldsymbol{v}_c)-1)\boldsymbol{u}_o - \sum_{k=1}^{K}(\sigma(-\boldsymbol{u}_k^T\boldsymbol{v}_c)-1)\boldsymbol{u}_k
\end{aligned}$$

$$\begin{aligned}
\frac{\partial J}{\partial \boldsymbol{u}_o} &= \frac{\partial}{\partial \boldsymbol{u}_o}((-log(\sigma(\boldsymbol{u}_o^T\boldsymbol{v}_c))) - \sum_{k=1}^{K} log(\sigma(-\boldsymbol{u}_k^T\boldsymbol{v}_c))) \\
&= \frac{-1}{\sigma(\boldsymbol{u}_o^T\boldsymbol{v}_c)}(\frac{\partial}{\partial \boldsymbol{u}_o}\sigma(\boldsymbol{u}_o^T\boldsymbol{v}_c)) - 0 \\
&= -(1-\sigma(\boldsymbol{u}_o^T\boldsymbol{v}_c))\frac{\partial}{\partial \boldsymbol{u}_o}(\boldsymbol{u}_o^T\boldsymbol{v}_c) \\
&= (\sigma(\boldsymbol{u}_o^T\boldsymbol{v}_c)-1)\boldsymbol{u}_o \\
\frac{\partial J}{\partial \boldsymbol{u}_k} &= \frac{\partial}{\partial \boldsymbol{u}_k}((-log(\sigma(\boldsymbol{u}_o^T\boldsymbol{v}_c))) - \sum_{k=1}^{K} log(\sigma(-\boldsymbol{u}_k^T\boldsymbol{v}_c))) \\
&= 0 - \frac{\partial}{\partial \boldsymbol{u}_k} log(\sigma(-\boldsymbol{u}_k^T\boldsymbol{v}_c)) \\
&= -\frac{1}{\sigma(-\boldsymbol{u}_k^T\boldsymbol{v}_c)}\sigma(-\boldsymbol{u}_k^T\boldsymbol{v}_c)(1-\sigma(-\boldsymbol{u}_k^T\boldsymbol{v}_c))\frac{\partial}{\partial \boldsymbol{u}_k}(-\boldsymbol{u}_k^T\boldsymbol{v}_c) \\
&= -(\sigma(-\boldsymbol{u}_k^T\boldsymbol{v}_c)-1)\boldsymbol{v}_c) \\
&= (\sigma(\boldsymbol{u}_k^T\boldsymbol{v}_c)\boldsymbol{v}_c, \text{ for all } k \neq o
\end{aligned}$$

Negative sampling is faster than softmax-CE loss at speed up ratio $\frac{V}{K}$, where V is all words in dictionary count, and K is the sampling size.

3.4 (d) Skip-gram and CBOW Gradients (8 pts)

Answer:

Derivatives for the skip-gram model

$$\frac{J_{skip-gram}(word_{c-m\ldots c+m})}{\partial \boldsymbol{v}_c} = \sum_{-m\leqslant j\leqslant m, j\neq 0} \frac{\partial F(\boldsymbol{w}_{c+j}, \boldsymbol{v}_c)}{\partial \boldsymbol{v}_c}$$

$$\frac{J_{skip-gram}(word_{c-m\ldots c+m})}{\partial \boldsymbol{v}_j} = 0, \forall j \neq c$$

$$\frac{J_{skip-gram}(word_{c-m\ldots c+m})}{\partial \boldsymbol{U}} = \sum_{-m\leqslant j\leqslant m, j\neq 0} \frac{\partial F(\boldsymbol{w}_{c+j}, \boldsymbol{v}_c)}{\partial \boldsymbol{U}}$$

Derivatives for the CBOW model

$$\frac{J_{CBOW}(word_{c-m\ldots c+m})}{\partial \boldsymbol{v}_j} = \frac{\partial F(\boldsymbol{w}_c, \hat{\boldsymbol{v}})}{\partial \hat{\boldsymbol{v}}}, \forall (j \neq c) \in \{c-m\ldots c+m\}$$

$$\frac{J_{CBOW}(word_{c-m\ldots c+m})}{\partial \boldsymbol{v}_j} = 0, \forall (j \neq c) \notin \{c-m\ldots c+m\}$$

$$\frac{J_{CBOW}(word_{c-m\ldots c+m})}{\partial \boldsymbol{U}} = \frac{\partial F(\boldsymbol{w}_c, \hat{\boldsymbol{v}})}{\partial \boldsymbol{U}}$$

3.5 (e) Word2Vec with SGD Code (12 pts)

Answer: See code: ~/`q3_word2vec.py`.

3.6 (f) SGD Code (4 pts)

Answer: See code: ~/`q3_sgd.py`.

3.7 (g) Train Vectors on Stanford Sentiment Treebank (4 pts)

Answer:

Explain: In the Word Vectors image, words clustered at similarity, such as the emotion words ”amazing”, ”wonderful” and ”great” are very close to each other. The word ”well” a little further but still close to ”amazing” , the connection characters and words ”the” ”a” ”,” etc are spread around alone.

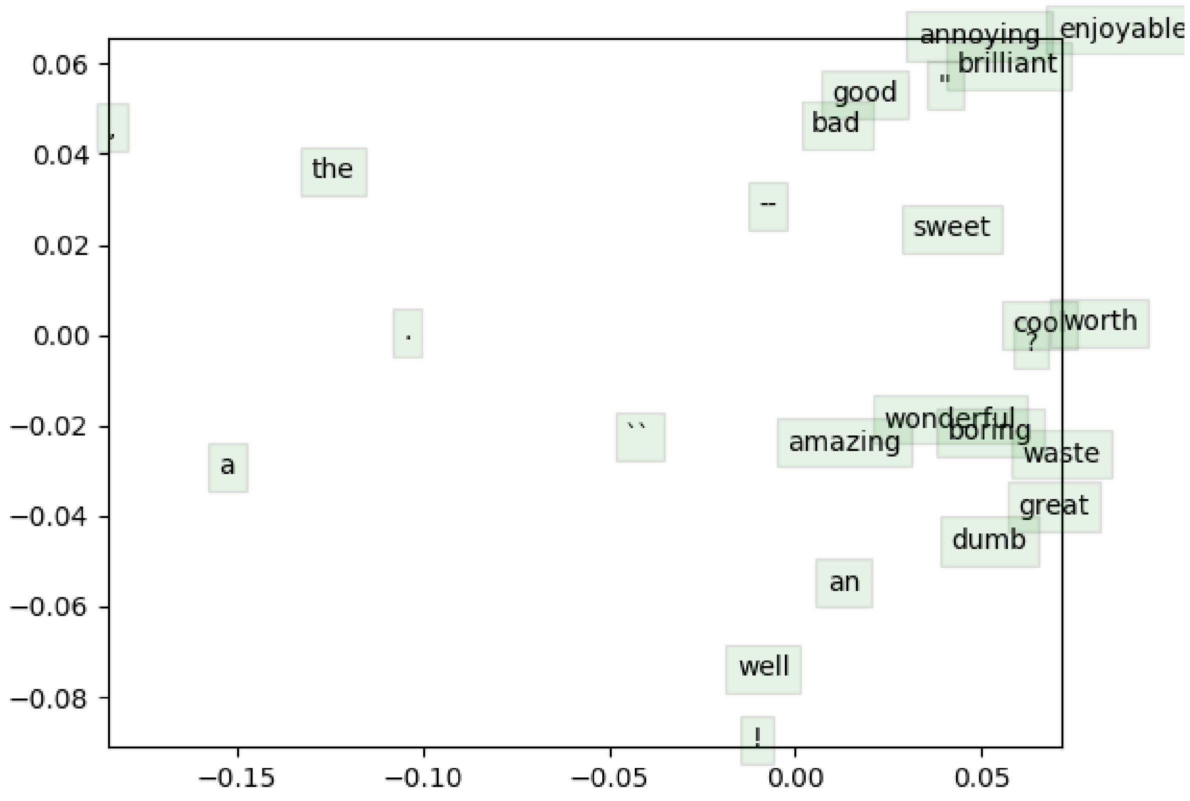


Figure 3.1: Word Vectors

3.8 (h) CBOW (Extra Credit: 2 pts)

Answer: See code: ~/`q3_sgd.py`.

Problem 4: Sentiment Analysis (20 pts)

4.1 (a) Sentence Features (2 pts)

Answer: See code: ~/q4_sentiment.py.

4.2 (b) Purpose of Regularization (1 pt)

Answer:

Purpose of regularization to constrain parameters, to avoid overfit on training dataset. And this can increase the generalizability of the model for new data.

4.3 (c) Purpose of Regularization (1 pt)

Answer: See code: ~/q4_sentiment.py.

The chooseBestModel code

```
### YOUR CODE HERE
best_dev = 0.0
for r in results:
    if r["dev"] > best_dev:
        bestResult = r
        best_dev = r["dev"]
### END YOUR CODE
```

4.4 (d) Pretrained vs New Vectors (3 pts)

Answer:

The Best train, dev and test accuracies using ”Your Vectors” are around $31.2\%, 30.7\%, 29.6\%$

Table 4.1: Sentiment Accuracy with different Regularization rate– ”Your Vectors” - on Stanford Sentiment Treebank dataset

Regularization Rate λ	**Accuracy**		
Use $\frac{1}{\lambda}$	*Train*	*Dev*	*Test*
0.000115	**31.262**	30.699	29.593
0.007920	31.203	**31.153**	**29.819**
0.011000	31.250	30.790	29.593
0.107000	30.466	30.881	29.276
1.050000	28.956	28.247	27.059

The Best train, dev and test accuracies using ”Pretrained” are around $39.97\%, 36.6\%, 37.3\%$

Table 4.2: Sentiment Accuracy with different Regularization rate – Pretrained GloVe

Regularization Rateλ	**Accuracy**		
Use $\frac{1}{\lambda}$	*Train*	*Dev*	*Test*
0.000115	39.923	36.331	37.014
0.000792	**39.970**	36.240	37.104
0.011000	39.899	36.240	37.195
0.107000	39.782	**36.603**	37.195
1.050000	39.478	36.512	**37.330**

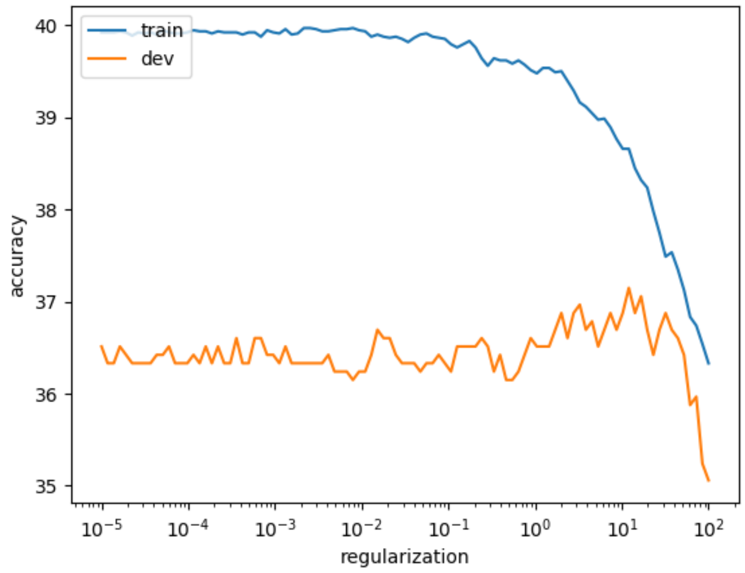
I think there below reasons making pretrained vectors did better

1. *GloVe vector has higher dimensions (*$D = 50$*)*
2. *Wikipedia dataset (billion tokens), which GloVe (pretrained) using is much larger than SST (215 thousands phrases)*
3. *GloVe use the global static information, while Word2Vec (skip-gram) not.*

4.5 (e) Accuracy vs. Regularization (4 pts)

Answer:

Figure 4.1: Accuracy vs. Regularization

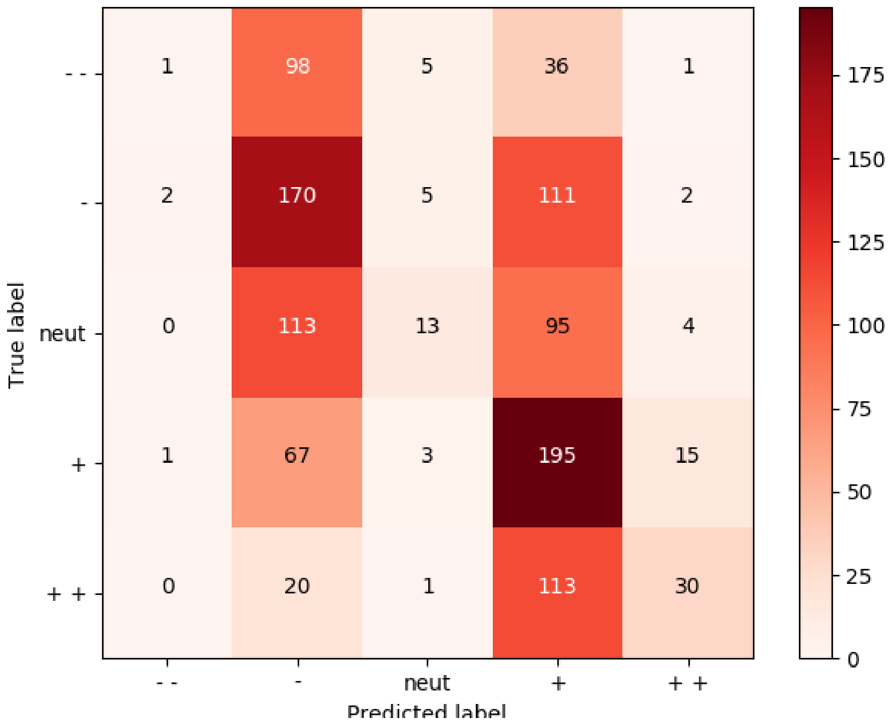


Explain:
In figure Accuracy vs. Regularization, the train accuracy and dev accuracy are better at $10^{-3}, 10^{-4}, and 10^{-5}$ regularization range, are around 40% and 36.3%. But the accuracy drops to 36% and 35% when regularization use $10^{1} and 10^{2}$. And also notice that the accuracy gap are around 4% for regularization $10^{-3}, 10^{-4}, and 10^{-5}$, which suggesting overfit on model, there is room to improve.

4.6 (f) Confusion Matrix (4 pts)

Answer:

Figure 4.2: Confusion Matrix (Development Set)



Explain:
Interpret of the Confusion Matrix
The larger number in diagonal the better. We can see the model hard to tell ”neutral” sentiment (value 13). And turn to put more ”neutral” to ”negative” (113 to 95). But model doesn’t make huge mitake, like put ”very negative” (–) to ”very positive” (++) or vice versa.

4.7 (g) Error Types & Features (4 pts)

Answer:

1. "`and if you 're not nearly moved to tears by a couple of scenes , you 've got ice water in your veins .`"
   - True $= +$; Predicted $= -$
   - Model not understand the idiom "moved to tears", so not understand whole sentence

2. "`it 's slow -- very , very slow . .`"
   - True $= --$; Predicted $= netural$
   - Model need to have more context know this is a very positive sentence".

3. "`nothing is sacred in this gut-buster . .`"
   - True $= -$; Predicted $= +$
   - Say something in converse way. That's common error makes by word-based model.