NLP - HW 1

Guy Tevet (305257206), Gavriel Habib (304946445)

Question 1 - Basics

(a) $softmax(x+c)_i = \frac{e^{x_i+c}}{\sum_j e^{x_j+c}} = \frac{e^c e^{x_i}}{\sum_j e^c e^{x_j}} = \frac{e^c e^{x_i}}{e^c \sum_j e^{x_j}} = \frac{e^{x_i}}{\sum_j e^{x_j}} = softmax(x)_i$

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

$$\frac{\partial \sigma}{\partial x} = \frac{e^{-x}}{(1+e^{-x})^2} = \frac{e^{-x}}{1+e^{-x}} \cdot \frac{1}{1+e^{-x}} = \left(1 - \frac{1}{1+e^{-x}}\right) \cdot \frac{1}{1+e^{-x}}$$
$$= \big(1-\sigma(x)\big) \cdot \sigma(x)$$

Question 2 – Word2Vec

Soft max loss function:

$$J = -\sum_{i=1}^{W} y_i \cdot \log\left(\frac{exp(u_i^T V_c)}{\sum_{w=1}^{W} exp(u_w^T V_c)}\right) = -\sum_{i=1}^{W} y_i u_i^T V_c + \sum_{i=1}^{W} y_i \cdot \log\Big(\sum_{w=1}^{W} exp(u_w^T V_c)\Big)$$

(a) $\frac{\partial J}{\partial V_c} = -\sum_{i=1}^{W} y_i u_i^T + \sum_{i=1}^{W} y_i \cdot \frac{1}{\sum_{w=1}^{W} exp(u_w^T V_c)} \sum_{w=1}^{W} u_w^T exp(u_w^T V_c)$

$= -u_k^T + \frac{1}{\sum_{w=1}^{W} exp(u_w^T V_c)} \sum_{w=1}^{W} u_w^T exp(u_w^T V_c)$

(b) $\frac{\partial J}{\partial u_{\hat{w}}} = -y_{\hat{w}} \cdot v_c + \sum_{i=1}^{W} y_i \cdot \frac{1}{\sum_{w=1}^{W} exp(u_w^T V_c)} \cdot v_c \exp\big(u_{\hat{w}}^T V_c\big)$

$$= -y_{\hat{w}} \cdot v_c + \frac{1}{\sum_{w=1}^{W} exp(u_w^T V_c)} \cdot v_c \exp\big(u_{\hat{w}}^T V_c\big)$$

Note that we used the fact that y is a one-hot vector ($y_k = 1$).

Negative sampling loss function:

$$J_{neg_{sample}}(o, v_c, U) = -\log\big(\sigma(u_0^T v_c)\big) - \sum_{k=1}^{K} \log\big(\sigma(-u_k^T v_c)\big)$$

(c) $\frac{\partial J}{\partial v_c} = -\frac{1}{\sigma(u_o^T v_c)} \sigma(u_o^T v_c)\big(1-\sigma(u_0^T v_c)\big)u_0^T$

$$-\sum_{k=1}^{K} \frac{1}{\sigma(-u_k^T v_c)} \sigma(-u_k^T v_c)\Big(1-\sigma(-u_k^T v_c)\Big)(-u_k^T)$$

$$= -\big(1-\sigma(u_0^T v_c)\big)u_0^T + \sum_{k=1}^{K} u_k^T \big(1-\sigma(-u_k^T v_c)\big)$$

$$\frac{\partial J}{\partial u_p} = v_c\Big(1-\sigma\big(-u_p^T v_c\big)\Big); \quad p \neq o$$

$$\frac{\partial J}{\partial u_o} = -v_c\big(1-\sigma(u_o^T v_c)\big)$$

(d) $L(c) = \sum_{\substack{-m \le j \le m \\ j \neq o}} F(w_{c+j}, v_c)$

$$\frac{\partial L(c)}{\partial w_{c+j}} = \sum_{\substack{-m \le j \le m \\ j \neq o}} \frac{\partial F(w_{c+j}, v_c)}{\partial w_{c+j}}$$

(h)run.py plot:

```
iter 39980: 9.524589
iter 39990: 9.550077
iter 40000: 9.577164
sanity check: cost at convergence should be around or below 10
training took 6393 seconds
Words related to "the":  ['the', 'if', 'that', 'comedy\\/thriller', 'or', 'a', '.', 'is', 'derek', 'bolt', 'decide']
Words related to "unique":  ['unique', '1979', 'puns', 'ba', 'realized', 'succumb', 'chabrolian', 'dares', 'regardless', 'imaginative', 'lunar']
Words related to "superb":  ['superb', 'mine', 'gold', 'zingers', 'moppets', 'roussillon', 'best', 'ghoulish', 'industry', 'pool', 'transporter']
Words related to "comedy":  ['comedy', 'sensation', 'observation', 'fast', 'first-timer', 'singing', 'cleaving', 'longest', 'cute', 'mature', 'often-funny']
Words related to "surprisingly":  ['surprisingly', 'either', 'hundred', '20-car', 'philandering', 'unusually', 'protective', 'dogs', 'bollywood', 'thinking', 'soderbergh']

Process finished with exit code 0
```

As can be seen, ‘the’ is closest to other conjunctions as ‘if’ and ‘that’ but also to not so related words such as ‘bolt’ and ‘decide’. More complicated words as ‘comedy’ is close to some way related words - ‘sensation’ , ‘fast’ , ‘singing’ in this case. However, it seems that the word ‘unique’ is closest to not related words.

Embeddings plot: