Natural Language Processing – HW1

Itay Itzhak – 305685877, Eytan Chamovitz – 203486550

Question 1

(a) Prove that $softmax(x) = softmax(x + c)$ for any input vector $x \in \mathbb{R}^d$ and constant c.

For a single dimension of the vector x:

$$softmax(x)_i = \frac{e^{x_i}}{\sum_{j=1}^{d} e^{x_j}}$$

And as such:

$$softmax(x+c)_i = \frac{e^{x_i+c}}{\sum_{j=1}^{d} e^{x_j+c}} = \frac{e^{x_i} \cdot e^{c}}{\sum_{j=1}^{d}(e^{x_j} \cdot e^{c})} = \frac{e^{x_i} \cdot e^{c}}{e^{c} \cdot \sum_{j=1}^{d}(e^{x_j})} = \frac{e^{x_i}}{\sum_{j=1}^{d} e^{x_j}} = softmax(x)_i$$

Because the same holds for every dimension, QED ■

(b) In code

(c) Derive the gradient of the Sigmoid function on a scalar:

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

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

(d) In code

(e) In code

Question 2

(a) Calculations:

Let $|W| \coloneqq \#\ words\ in\ vocab$, $\boldsymbol{y}, \hat{\boldsymbol{y}} \in \mathbb{R}^{|W|\times 1}$, $u_w, v_c \in \mathbb{R}^{D\times 1}$, $D \coloneqq embedding\ dimension$, the vector y is 1-hot encoded ($\exists k\ s.t. \forall i \neq k\ y_i = 0, y_k = 1$), $U \in \mathbb{R}^{D\times|W|}$ is the matrix whose columns are u_w and has D rows.

$$\hat{y}_o = \Pr(o|c) = \frac{e^{(u_o^T \cdot v_c)}}{\sum_{w=1}^{|W|} e^{(u_w^T \cdot v_c)}}$$

The loss function:

$$J(\boldsymbol{y},\hat{\boldsymbol{y}}) = -\sum_{i=1}^{|W|} y_i \cdot \log(\hat{y}_i) = -\sum_{i=1}^{|W|}\left(y_i \cdot \log\left(\frac{e^{(u_i^T \cdot v_C)}}{\sum_{w=1}^{|W|} e^{(u_w^T \cdot v_C)}}\right)\right)$$
$$= -\sum_{i=1}^{|W|}\left(y_i \cdot \left[u_i^T \cdot v_C - \log\left(\sum_{w=1}^{|W|} e^{(u_w^T \cdot v_C)}\right)\right]\right)$$

And because of the 1-hot encoding (let k be the index of the 1):

$$J(\boldsymbol{y},\hat{\boldsymbol{y}}) = -\left[u_k^T \cdot v_C - \log\left(\sum_{w=1}^{|W|} e^{(u_w^T \cdot v_C)}\right)\right]$$

Solving for $\frac{\partial J}{\partial v_c}$:

$$\frac{\partial J}{\partial v_c} = -\left[u_k - \frac{1}{\sum_{j=1}^{|W|} e^{\left(u_j^T \cdot v_C\right)}} \cdot \sum_{w=1}^{|W|} e^{(u_w^T \cdot v_c)} \cdot u_w\right] = \sum_{w=1}^{|W|}\left(\frac{e^{(u_w^T \cdot v_c)}}{\sum_{j=1}^{|W|} e^{\left(u_j^T \cdot v_C\right)}} \cdot u_w\right) - u_k$$
$$= \sum_{w=1}^{|W|} (\hat{y}_w \cdot u_w) - u_k$$

Because y is 1-hot with 1 in position k, then $u_k = U \cdot \boldsymbol{y}$. In addition, $\sum_{w=1}^{|W|} \hat{y}_w \cdot u_w = U \cdot \hat{\boldsymbol{y}}$

And in total we got:

$$\frac{\partial J}{\partial v_c} = U[\hat{\boldsymbol{y}} - \boldsymbol{y}]$$

(b) Calculations:

The function:

$$CE(y,\hat{y}) = -\sum_o y_o \cdot \log(\hat{y}_o)$$

Applying the Chain Rule:

$$\frac{\partial}{\partial u_w} CE(y,\hat{y}) = -\sum_o y_o \cdot \frac{\partial \log \hat{y}_o}{\partial u_w} = -\sum_o y_o \cdot \frac{1}{\hat{y}_o} \cdot \frac{\partial \hat{y}_o}{\partial u_w}$$

Calculate the inner derivative:

$$\hat{y}_o = \frac{\exp(u_o^T \cdot v_c)}{\sum_{k=1}^{W} \exp(u_k^T \cdot v_c)}$$

$$\frac{\partial \hat{y}_o}{\partial u_w} = \frac{\frac{\partial}{\partial u_w}\exp(u_o^T \cdot v_c) \cdot \sum_{k=1}^{W}\left(\exp\left(u_k^T \cdot v_c\right)\right) - \exp(u_o^T \cdot v_c) \cdot \frac{\partial}{\partial u_w}\sum_{k=1}^{W}\left(\exp\left(u_k^T \cdot v_c\right)\right)}{\left(\sum_{k=1}^{W}(\exp(u_k^T \cdot v_c))\right)^2}$$

$$= \begin{cases} \dfrac{\exp(u_w^T \cdot v_c) \cdot v_c \cdot \sum_{k=1}^{W}\left(\exp\left(u_k^T \cdot v_c\right)\right) - \exp(u_o^T \cdot v_c) \cdot \exp(u_w^T \cdot v_c)}{\left(\sum_{k=1}^{W}(\exp(u_k^T \cdot v_c))\right)^2} & w = o \\ \dfrac{0 - \exp(u_o^T \cdot v_c) \cdot \exp(u_w^T \cdot v_c) \cdot v_c}{\left(\sum_{k=1}^{W}(\exp(u_k^T \cdot v_c))\right)^2} & w \neq o \end{cases}$$

$$= \begin{cases} \dfrac{\exp(u_w^T \cdot v_c)}{\sum_{k=1}^{W}(\exp(u_k^T \cdot v_c))} \cdot \dfrac{\left(\sum_{k=1}^{W}\left(\exp\left(u_k^T \cdot v_c\right)\right) - \exp(u_o^T \cdot v_c)\right)}{\sum_{k=1}^{W}(\exp(u_k^T \cdot v_c))} \cdot v_c & w = o \\ \dfrac{-\exp(u_o^T \cdot v_c)}{\sum_{k=1}^{W}(\exp(u_k^T \cdot v_c))} \cdot \dfrac{\exp(u_w^T \cdot v_c)}{\sum_{k=1}^{W}(\exp(u_k^T \cdot v_c))} \cdot v_c & w \neq o \end{cases}$$

$$= \begin{cases} \hat{y}_w \cdot (1 - \hat{y}_w) \cdot v_c & w = o \\ -\hat{y}_o \cdot \hat{y}_w \cdot v_c & w \neq o \end{cases}$$

$$\Longrightarrow \frac{\partial}{\partial u_w} CE(y, \hat{y}) = -\sum_o y_o \cdot \frac{1}{\hat{y}_o} \cdot \frac{\partial \hat{y}_o}{\partial u_w} = -y_w \cdot \frac{1}{\hat{y}_w} \cdot \hat{y}_W \cdot (1 - \hat{y}_w) \cdot v_c - \sum_{o \neq w} y_o \cdot \frac{1}{\hat{y}_o} \cdot -\hat{y}_o \cdot \hat{y}_w \cdot v_c$$

$$= -y_w \cdot (1 - \hat{y}_w) \cdot v_c + \hat{y}_w \cdot \sum_{o \neq w} y_o \cdot v_c = \left(-y_w + y_w \cdot \hat{y}_w + \hat{y}_w \cdot \sum_{o \neq w} y_o\right) \cdot v_c$$

$$= \left(\hat{y}_w \cdot \left(y_w + \sum_{o \neq w} y_o\right) - y_w\right) \cdot v_c \xrightarrow[\substack{\sum_o y_o = 1 \\ y \text{ is 1 hot encoded}}]{} \frac{\partial}{\partial u_w} CE(y, \hat{y}) = (\hat{y}_w - y_w) \cdot v_c$$

(c) Calculations:

The function:

$$J_{neg-sample}(o, v_c, U) = -\log\left(\sigma(u_O^T v_c)\right) - \sum_{k=1}^{K} \log(\sigma(-u_k^T v_c))$$

Derivative according to v_c:

Applying the Chain Rule to each dimension separately:

$$\frac{\partial}{\partial v_c} J_{neg-sample}(o, v_c, U) = -\frac{1}{\sigma(u_O^T v_c)} \cdot \frac{\partial \sigma(u_O^T v_c)}{\partial v_c} - \sum_{k=1}^{K} \frac{1}{\sigma(-u_k^T v_c)} \cdot \frac{\partial \sigma(-u_k^T v_c)}{\partial v_c}$$

Calculate the inner derivative:

$$\frac{\partial \sigma(u_O^T v_c)}{\partial v_c} = \frac{\partial}{\partial v_c}\left(1 + e^{-u_O^T v_c}\right)^{-1} = -\left(1 + e^{-u_O^T v_c}\right)^{-2} \frac{\partial}{\partial v_c}\left(1 + e^{-u_O^T v_c}\right)$$

$$= -\left(1 + e^{-u_O^T v_c}\right)^{-2} e^{-u_O^T v_c} \cdot -u_O^T = \frac{e^{-u_O^T v_c} \cdot u_O^T}{\left(1 + e^{-u_O^T v_c}\right)^2} = \frac{e^{-u_O^T v_c} + 1 - 1}{1 + e^{-u_O^T v_c}} \cdot \frac{u_O^T}{1 + e^{-u_O^T v_c}}$$

$$= \left(1 - \sigma(u_O^T v_c)\right) \cdot \sigma(u_O^T v_c) \cdot u_O^T$$

$$\frac{\partial \sigma(-u_k^T v_c)}{\partial v_c} = \left(1 - \sigma(-u_k^T v_c)\right) \cdot \sigma(-u_k^T v_c) \cdot -u_k^T$$

Combine for each dimension:

$$\Longrightarrow \frac{\partial}{\partial v_c} J_{neg-sample}(o, v_c, U)$$

$$= -\frac{1}{\cancel{\sigma(u_O^T v_c)}} \cdot \left(1 - \sigma(u_O^T v_c)\right) \cdot \cancel{\sigma(u_O^T v_c)} \cdot u_O^T$$

$$-\sum_{k=1}^{K} \frac{1}{\cancel{\sigma(-u_k^T v_c)}} \cdot \left(1 - \sigma(-u_k^T v_c)\right) \cdot \cancel{\sigma(-u_k^T v_c)} \cdot -u_k^T$$

$$= -\left(1 - \sigma(u_O^T v_c)\right) \cdot \boldsymbol{u_O^T} + \sum_{k=1}^{K} \left(1 - \sigma(-u_k^T v_c)\right) \cdot \boldsymbol{u_k^T}$$

<u>Derivative according to u_j:</u>

Applying the Chain Rule to each dimension separately:

$$\frac{\partial}{\partial u_j} J_{neg-sample}(o, v_c, U) = -\frac{1}{\sigma(u_O^T v_c)} \cdot \frac{\partial \sigma(u_O^T v_c)}{\partial u_j} - \sum_{k=1}^{K} \frac{1}{\sigma(-u_k^T v_c)} \cdot \frac{\partial \sigma(-u_k^T v_c)}{\partial u_j}$$

Calculate the inner derivative:

If $j = o$:

$$\frac{\partial \sigma(u_O^T v_c)}{\partial u_O} = \left(1 - \sigma(u_O^T v_c)\right) \cdot \sigma(u_O^T v_c) \cdot v_c, \qquad \frac{\partial \sigma(-u_k^T v_c)}{\partial u_O} = 0$$

If $j \neq o$:

$$\frac{\partial \sigma(u_O^T v_c)}{\partial u_j} = 0, \qquad \frac{\partial \sigma(-u_k^T v_c)}{\partial u_j} = \begin{cases} \left(1 - \sigma(-u_j^T v_c)\right) \cdot \sigma(-u_j^T v_c) \cdot -v_c & k = j \\ 0 & k \neq j \end{cases}$$

Derivative for $\frac{\partial J}{\partial u_o}$:

$$\frac{\partial}{\partial u_O} J_{neg-sample}(o, v_c, U) = -\frac{1}{\cancel{\sigma(u_O^T v_c)}} \cdot \left(1 - \sigma(u_O^T v_c)\right) \cdot \cancel{\sigma(u_O^T v_c)} \cdot v_c - 0 = (\sigma(u_O^T v_c) - 1) \cdot \boldsymbol{v_c}$$

Derivative for $\frac{\partial J}{\partial u_j}$ and $j \neq o$:

$$\frac{\partial}{\partial u_j} J_{neg-sample}(o, v_c, U) = 0 - \frac{1}{\cancel{\sigma(-u_k^T v_c)}} \cdot \left(1 - \sigma(-u_j^T v_c)\right) \cdot \cancel{\sigma(-u_j^T v_c)} \cdot -v_c$$
$$= \left(1 - \sigma(-u_j^T v_c)\right) \cdot v_c = \sigma(u_j^T v_c) \cdot v_c = \left(\sigma(u_j^T v_c) - 0\right) \cdot \boldsymbol{v_c}$$

Thus, we can look at the matrix notation:

$$\frac{\partial}{\partial U} J_{neg-sample}(o, v_c, U) = \left(\sigma(U \cdot \boldsymbol{v_c}) - \boldsymbol{e_1^K}\right) \cdot \boldsymbol{v_c}$$

Where e_1^K is a column vector of length K with 1 in the first entry and 0 everywhere else.

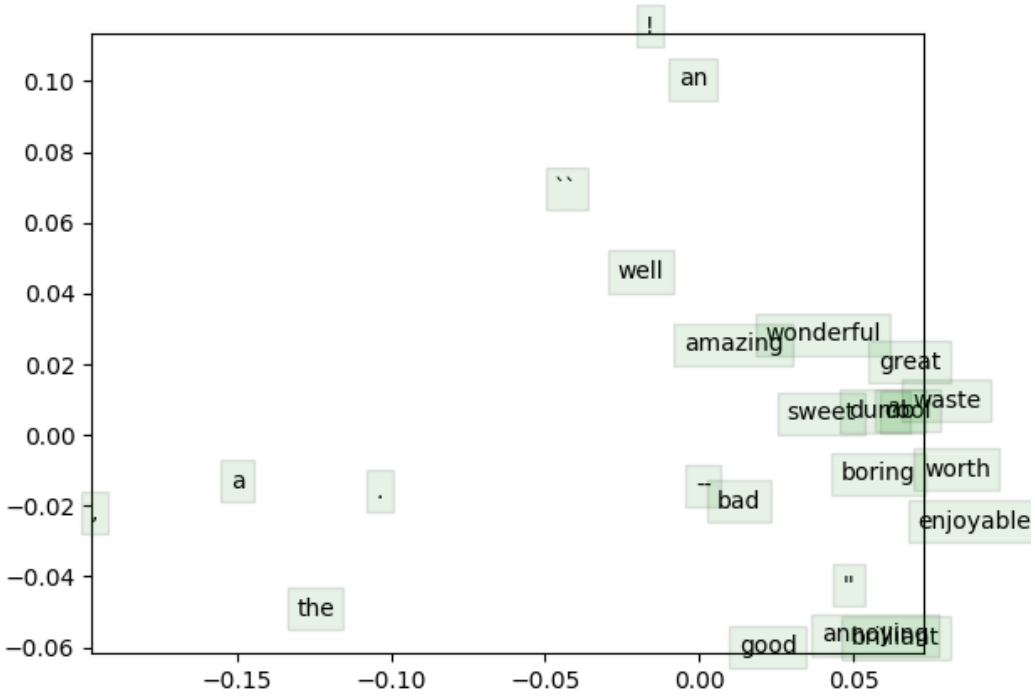
(d) Calculations

We'll need to derive gradients with respect to the negative sample words ($k \in K$) and the center word.

$$\frac{\partial}{\partial w_k} \sum_{-m \leq j \leq m, j \neq 0} F(w_{c+j}, v_c) = \sum_{-m \leq j \leq m, j \neq 0} \frac{\partial}{\partial w_k} F(w_{c+j}, v_c) \cdot I(k, c+j)$$

$$\frac{\partial}{\partial v_c} \sum_{-m \leq j \leq m, j \neq 0} F(w_{c+j}, v_c) = \sum_{-m \leq j \leq m, j \neq 0} \frac{\partial}{\partial v_c} F(w_{c+j}, v_c)$$

(e) In code
(f) In code
(g) In code
(h) Picture




```
Words related to "the":  ['decide', 'bolt', '.', 'is', 'derek', 'a', 'if', 'that', 'comedy\\/thriller', 'or',
'the']
Words related to "unique":  ['lunar', 'imaginative', 'chabrolian', 'regardless', 'dares', 'succumb', 'ba',
'1979', 'puns', 'realized', 'unique']
Words related to "superb":  ['transporter', 'pool', 'industry', 'best', 'ghoulish', 'roussillon', 'zingers',
'gold', 'moppets', 'mine', 'superb']
Words related to "comedy":  ['often-funny', 'mature', 'singing', 'observation', 'cute', 'fast', 'cleaving',
'longest', 'first-timer', 'sensation', 'comedy']
Words related to "surprisingly":  ['soderbergh', 'thinking', 'bollywood', 'dogs', 'protective', 'unusually',
'philandering', '20-car', 'hundred', 'either', 'surprisingly']
```

It's possible to see that words that are close in their meaning as "wonderful" and "amazing" are close, as well as words that are used in similar sentences parts as "a" and "the". The idea of similar words have close vectors representations as talked about in class demonstrated here in visual and in the list of close words. The results are worse than the word2vec that we saw in the notebook on question 3 problebly due to tweaks in algorithm and huge amounts of data that were used.

Question 3

(a) Polysemous words:

The word we chose is "right" which can have meanings of both "direction", "correctness", "morality", and "law" (as in legal/civil rights). The results that we got are:

```
[('Right', 0.5703941583633423),
 ('wrong', 0.5534271001815796),
```

```
('##.Help_us', 0.5502839684486389),
('Goodwill_Catanese', 0.5159174799919128),
('left', 0.49213987588882446),
('fielder_Joe_Borchard', 0.48948538303375244),
('NOTE_Xactly_Incent', 0.48866304755210876),
('fielder_Ambiorix_Concepcion', 0.4841775894165039),
('now', 0.4794555902481079),
('fielder_Jeromy_Burnitz', 0.47718381881713867)]
```

As can be seen, we have words that relate both to the "correctness" meaning (wrong), and the "directions/side" meaning (left, and names of Right Fielders in Baseball).

The previous word that we tried was "rose", as in "going up", "flower" or a Name, but all were related only to the "going up" meaning. This is probably due to the popularity of this context for the word in the news. (There are probably more news stories regarding rising stocks, climbing, increasing process, etc. than there are of people named Rose or of flowers).

(b) Close antonyms and far synonyms

The words we chose were "good", "wonderful", and "bad". The results we got were:

```
Synonyms good, wonderful have cosine distance: 0.4273882508277893
Antonyms good, bad have cosine distance: 0.28099489212036133
```

The distance between "good" and "wonderful" is because "good" can be used in much wider contexts than "wonderful", which can mostly be used as an adjective. "bad" has similar general uses as "good" does, so they will be closer.

These were the only words we tried. ☺

(c) Analogies:

Our successful analogies are:

`Pittsburgh:Steelers::Seattle:Seahawks` and `Trump:USA::Netanyahu:Israel`

The top-3 results are:

```
('Seahawks', 0.7275974750518799),
('Seattle_Seahawks', 0.673957109451294),
('Niners', 0.6002556085586548)
```

And:

```
('Israel', 0.5105854868888855),
('Prime_Minister_Benjamin_Netanyahu', 0.45393186807632446),
('Prime_Minister_Binyamin_Netanyahu', 0.4437536299228668)
```

Just to clarify the first analogy, the Steelers are the football team from Pittsburgh, and the Seahawks are the football team from Seattle. We don't think the Trump-Bibi analogy needs explaining.

(d) Analogy:

Our successful analogies are:

```
Lion:lioness::stag:doe
```

The top-3 results are:

```
('friend_Guy_Pelly', 0.4229981601238251),
 ('hour_booze_bender', 0.4157106280326843),
 ('stag_dos', 0.40979689359664917),
```

The analogy intended was for the male and female forms of the different animals, but did not work due the extensive use of the word “stag” in American culture as an “alpha male” man.