CS 224n Assignment #2: word2vec (44 Points)

Carolyn Kao (#06366163) chkao831@stanford.edu

Jan 26, 2021

1 Written: Understanding word2vec (26 points)

Let's have a quick refresher on the `word2vec` algorithm. The key insight behind `word2vec` is that *'a word is known by the company it keeps'*. Concretely, suppose we have a 'center' word c and a contextual window surrounding c. We shall refer to words that lie in this contextual window as 'outside words'. For example, in Figure 1 we see that the center word c is 'banking'. Since the context window size is 2, the outside words are 'turning', 'into', 'crises', and 'as'.

The goal of the skip-gram `word2vec` algorithm is to accurately learn the probability distribution $P(O|C)$. Given a specific word o and a specific word c, we want to calculate $P(O = o|C = c)$, which is the probability that word o is an 'outside' word for c, i.e., the probability that o falls within the contextual window of c.

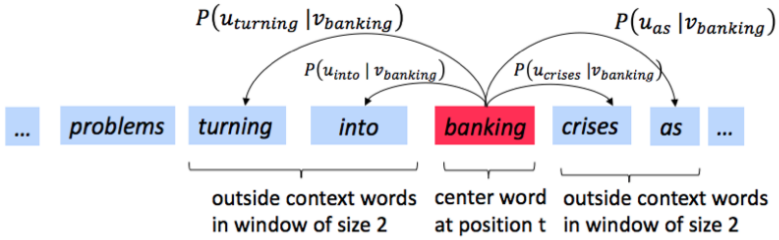


Figure 1: The word2vec skip-gram prediction model with window size 2

In `word2vec`, the conditional probability distribution is given by taking vector dot-products and applying the softmax function:

$$P(O = o \mid C = c) = \frac{\exp(\boldsymbol{u}_o^\top \boldsymbol{v}_c)}{\sum_{w \in \text{Vocab}} \exp(\boldsymbol{u}_w^\top \boldsymbol{v}_c)} \tag{1}$$

Here, $\boldsymbol{u}_o$ is the 'outside' vector representing outside word o, and $\boldsymbol{v}_c$ is the 'center' vector representing center word c. To contain these parameters, we have two matrices, $\boldsymbol{U}$ and $\boldsymbol{V}$. The columns of $\boldsymbol{U}$ are all the 'outside' vectors $\boldsymbol{u}_w$. The columns of $\boldsymbol{V}$ are all of the 'center' vectors $\boldsymbol{v}_w$. Both $\boldsymbol{U}$ and $\boldsymbol{V}$ contain a vector for every $w \in$ Vocabulary.[1]

u_K: Kth column of U; the outside vector for the word indexed by K
V_K: Kth column of V; the center vector for the word indexed by K

Recall from lectures that, for a single pair of words c and o, the loss is given by:

$$\boldsymbol{J}_{\text{naive-softmax}}(\boldsymbol{v}_c, o, \boldsymbol{U}) = -\log P(O = o|C = c). \tag{2}$$

We can view this loss as the cross-entropy[2] between the true distribution $\boldsymbol{y}$ and the predicted distribution $\hat{\boldsymbol{y}}$. Here, both $\boldsymbol{y}$ and $\hat{\boldsymbol{y}}$ are vectors with length equal to the number of words in the vocabulary. Furthermore, the k^{th} entry in these vectors indicates the conditional probability of the k^{th} word being an 'outside word' for the given c. The true empirical distribution $\boldsymbol{y}$ is a one-hot vector with a 1 for the true outside word o, and 0 everywhere else. The predicted distribution $\hat{\boldsymbol{y}}$ is the probability distribution $P(O|C = c)$ given by our model in equation (1).

(a) (3 points) Show that the naive-softmax loss given in Equation (2) is the same as the cross-entropy loss between $\boldsymbol{y}$ and $\hat{\boldsymbol{y}}$; i.e., show that

$$-\sum_{w \in Vocab} y_w \log(\hat{y}_w) = -\log(\hat{y}_o). \tag{3}$$

Your answer should be one line.

$\because$ y is a one-hot vector in which $\begin{cases} y_w = 1 & \text{when } w = o \\ y_w = 0 & \text{when } w \neq o \end{cases}$

$\therefore$ $-\sum_{w \in Vocab} y_w \log(\hat{y}_w)$

$= -[y_1 \log \hat{y}_1 + y_2 \log \hat{y}_2 + \cdots + y_o \log \hat{y}_o + \cdots + y_w \log \hat{y}_w]$

$= -y_o \log \hat{y}_o$

$= -\log \hat{y}_o$ #

(b) (5 points) Compute the partial derivative of $\boldsymbol{J}_{\text{naive-softmax}}(\boldsymbol{v}_c, o, \boldsymbol{U})$ with respect to $\boldsymbol{v}_c$. Please write your answer in terms of $\boldsymbol{y}$, $\hat{\boldsymbol{y}}$, and $\boldsymbol{U}$. Note that in this course, we expect your final answers to follow the shape convention.[3] This means that the partial derivative of any function $f(x)$ with respect to x should have the same shape as x. For this subpart, please present your answer in vectorized form. In particular, you may not refer to specific elements of $\boldsymbol{y}$, $\hat{\boldsymbol{y}}$, and $\boldsymbol{U}$ in your final answer (such as $\boldsymbol{y}_1$, $\boldsymbol{y}_2$, ...).

Method I: Presented in Matrix-Vector Multiplication form

From (a), we see that the naive-softmax loss function can be expressed as cross-entropy (CE) loss btw y and $\hat{y}$.

Hence denote $J = CE(y, \hat{y})$

Then, denote the input vector as $\theta = U^T v_c$ and the predicted function $\hat{y}$ as function of θ.

$$\frac{\partial J}{\partial \theta} = (\hat{y} - y)^T$$

(ref: http://courses.cs.ut.ee/MTAT.03.277/2015_fall/uploads/Main/word2vec.pdf)

Using chain rule,

$$\frac{\partial J}{\partial v_c} = \frac{\partial J}{\partial \theta} \frac{\partial \theta}{\partial v_c}$$

$$= (\hat{y} - y)^T \frac{\partial U^T v_c}{\partial v_c}$$

$$= U(\hat{y} - y) \quad \#$$

(Akshay's OH on Nook)

Method II: Presented in Vector-Vector Addition Form

$$J(v_c, o, U) = -\log(\hat{y}_o)$$

$$= -\log\left(\frac{\exp(u_o^T v_c)}{\sum_{w \in Vocab} \exp(u_w^T v_c)}\right)$$

$$= -\left[\log(\exp(u_o^T v_c)) - \log\left(\sum_{w'} \exp u_{w'}^T v_c\right)\right]$$

$$= -u_o^T v_c + \log\left(\sum_{w'} \exp(u_{w'}^T v_c)\right)$$

$$\frac{\partial J}{\partial v_c} = -u_o + \frac{1}{\sum_{w'} \exp(u_{w'}^T v_c)} \frac{\partial}{\partial v_c} \sum_{w} \exp(u_w^T v_c)$$

$$= -u_o + \frac{1}{\sum_{w'} \exp(u_{w'}^T v_c)} \sum_w \frac{\partial}{\partial v_c} \exp(u_w^T v_c)$$

$$= -u_o + \frac{1}{\sum_{w'} \exp(u_{w'}^T v_c)} \sum_w \exp(u_w^T v_c) u_w$$

$$= -u_o + \sum_w \frac{\exp(u_w^T v_c)}{\sum_{w'} \exp(u_{w'}^T v_c)} u_w$$

$$= \underbrace{\sum_w \hat{y}_w u_w}_{\text{It is the expected context word}} - \underbrace{u_o}_{\text{It is the observed representation of the outside word}} \quad \#$$

(c) (5 points) Compute the partial derivatives of $\boldsymbol{J}_{\text{naive-softmax}}(\boldsymbol{v}_c, o, \boldsymbol{U})$ with respect to each of the 'outside' word vectors, $\boldsymbol{u}_w$'s. There will be two cases: when $w = o$, the true 'outside' word vector, and $w \neq o$, for all other words. Please write your answer in terms of $\boldsymbol{y}$, $\hat{\boldsymbol{y}}$, and $\boldsymbol{v}_c$. In this subpart, you may use specific elements within these terms as well, such as $(\boldsymbol{y}_1, \boldsymbol{y}_2, \dots)$.

(Akshay's OH on Nook)

$$J(v_c, o, U) = -\log(\hat{y}_o)$$

$$= -\log\left(\frac{\exp(u_o^T v_c)}{\sum_{w \in \text{Vocab}} \exp(u_w^T v_c)}\right)$$

$$= -\left[\log(\exp(u_o^T v_c)) - \log\left(\sum_{w'} \exp u_{w'}^T v_c\right)\right]$$

$$= -u_o^T v_c + \log\left(\sum_{w'} \exp(u_{w'}^T v_c)\right)$$

take derivative of this

$$\frac{\partial}{\partial u_w} \log\left(\sum_{w'} \exp(u_{w'}^T v_c)\right)$$

$$= \frac{1}{\sum_{w'} \exp(u_{w'}^T v_c)} \frac{\partial}{\partial u_w} \sum_{w} \exp(u_{w'}^T v_c)$$

$$= \frac{1}{\sum_{w'} \exp(u_{w'}^T v_c)} \frac{\partial}{\partial u_w} \exp(u_w^T v_c)$$

$$= \underbrace{\left[\frac{\exp(u_w^T v_c)}{\sum_{w'} \exp(u_{w'}^T v_c)}\right]}_{P(W=w \mid C=c)} v_c = \hat{y}_w v_c$$

If $w = o$, $\partial J / \partial v_w = [P(W=w \mid C=c) - 1] v_c$

If $w \neq o$, $\partial J / \partial v_w = P(W=w \mid C=c) v_c$

Hence, $\dfrac{\partial J_{\text{naive-softmax}}(v_c, o, U)}{\partial v_w} = (\hat{y} - y)^T v_c = \hat{y}_w v_c - y_w v_c$ #

(d) (1 point) Compute the partial derivative of $\boldsymbol{J}_{\text{naive-softmax}}(\boldsymbol{v}_c, o, \boldsymbol{U})$ with respect to U. Please write your answer in terms of $\frac{\partial \boldsymbol{J}(\boldsymbol{v}_c,o,\boldsymbol{U})}{\partial \boldsymbol{u}_1}$, $\frac{\partial \boldsymbol{J}(\boldsymbol{v}_c,o,\boldsymbol{U})}{\partial \boldsymbol{u}_2}$, $\ldots$, $\frac{\partial \boldsymbol{J}(\boldsymbol{v}_c,o,\boldsymbol{U})}{\partial \boldsymbol{u}_{|Vocab|}}$. The solution should be one or two lines long.

(Rui Wang's OH on Nook)

$U = \Big[\overbrace{\quad\quad}^{N} \Big]$ } dim = M by shape convention,

$\frac{\partial J}{\partial U}$ is M x N

$$\frac{\partial J}{\partial U} = \left[\frac{\partial J}{\partial u_1}, \frac{\partial J}{\partial u_2}, \ldots, \frac{\partial J}{\partial u_{|Vocab|}} \right] \#$$

(This is a row matrix formed by column vec's, each w/ dim M)

(e) (3 Points) The sigmoid function is given by Equation 4:

$$\sigma(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{e^x+1} \tag{4}$$

Please compute the derivative of $\sigma(x)$ with respect to x, where x is a scalar. Hint: you may want to write your answer in terms of $\sigma(x)$.

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

$$= \frac{(e^x+1)e^x - e^x e^x}{(e^x+1)^2}$$

$$= \frac{e^x}{(e^x+1)^2}$$

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

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

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

$$= \sigma(x)\,(1-\sigma(x)) \quad \#$$

(f) (4 points) Now we shall consider the Negative Sampling loss, which is an alternative to the Naive Softmax loss. Assume that K negative samples (words) are drawn from the vocabulary. For simplicity of notation we shall refer to them as $w_1, w_2, \ldots, w_K$ and their outside vectors as $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_K$. For this question, assume that the K negative samples are distinct. In other words, $i \neq j$ implies $w_i \neq w_j$ for $i, j \in \{1, \ldots, K\}$. Note that $o \notin \{w_1, \ldots, w_K\}$. For a center word c and an outside word o, the negative sampling loss function is given by:

$$\boldsymbol{J}_{\text{neg-sample}}(\boldsymbol{v}_c, o, \boldsymbol{U}) = -\log(\sigma(\boldsymbol{u}_o^\top \boldsymbol{v}_c)) - \sum_{k=1}^{K} \log(\sigma(-\boldsymbol{u}_k^\top \boldsymbol{v}_c)) \tag{5}$$

for a sample $w_1, \ldots w_K$, where $\sigma(\cdot)$ is the sigmoid function.[4]

Please repeat parts (b) and (c), computing the partial derivatives of $\boldsymbol{J}_{\text{neg-sample}}$ with respect to $\boldsymbol{v}_c$, with respect to $\boldsymbol{u}_o$, and with respect to a negative sample $\boldsymbol{u}_k$. Please write your answers in terms of the vectors $\boldsymbol{u}_o$, $\boldsymbol{v}_c$, and $\boldsymbol{u}_k$, where $k \in [1, K]$. After you've done this, describe with one sentence why this loss function is much more efficient to compute than the naive-softmax loss. Note, you should be able to use your solution to part (e) to help compute the necessary gradients here.

$$J = -\log(\sigma(u_o^T v_c)) - \sum_{k=1}^{K} \log(\sigma(-u_k^T v_c))$$

with respect to v_c

$$\frac{\partial J}{\partial v_c} = \frac{\partial}{\partial v_c}\left[-\log(\sigma(u_o^T v_c)) - \sum_{k=1}^{K} \log(\sigma(-u_k^T v_c))\right]$$

$$= \frac{\partial}{\partial v_c}\left[-\log(\sigma(u_o^T v_c))\right] - \frac{\partial}{\partial v_c}\left[\sum_{k=1}^{K} \log(\sigma(-u_k^T v_c))\right]$$

$$= -u_o(1 - \sigma(u_o^T v_c)) - \left[-\sum_{k=1}^{K} u_k(1 - \sigma(-u_k^T v_c))\right]$$

$$= (\sigma(u_o^T v_c) - 1)\, u_o - \sum_{k=1}^{K} (\sigma(-u_k^T v_c) - 1)\, u_k \quad \#$$

with respect to u_o

$$\frac{\partial J}{\partial u_o} = \frac{\partial}{\partial u_o}\left[-\log(\sigma(u_o^T v_c)) - \sum_{k=1}^{K} \log(\sigma(-u_k^T v_c))\right]$$

$$= \frac{\partial}{\partial u_o}\left[-\log(\sigma(u_o^T v_c))\right] - 0$$

$$= -\left\{\frac{1}{\sigma(u_o^T v_c)}\left[\sigma(u_o^T v_c)(1 - \sigma(u_o^T v_c))\, v_c\right]\right\}$$

$$= (\sigma(u_o^T v_c) - 1)\, v_c$$

with respect to u_K

∵ All words are distinct

∴ $u_1, u_2 \ldots$ each corresponds to a specific word —

Do on $-\sum_{i \in \{1,\ldots,K \text{ where } w_i = w_K\}} \log(\sigma(-u_i^T v_c))$

$$\Rightarrow \frac{\partial J}{\partial u_K} = -\frac{\partial}{\partial u_K}\left[\sum_{K=1}^{K} \log(\sigma(-u_K^T v_c))\right]$$

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

$$= \frac{\sigma(-u_K^T v_c)(1-\sigma(-u_K^T v_c))v_c}{\sigma(-u_K^T v_c)}$$

$$= (1-\sigma(-u_K^T v_c))v_c \quad \#$$

Negative Sampling loss only computes a fixed size of the outside vectors U instead on the whole thing as in naive softmax loss; Hence, its computationally more efficient.

(g) (2 point) Now we will repeat the previous exercise, but without the assumption that the K sampled words are distinct. Assume that K negative samples (words) are drawn from the vocabulary. For simplicity of notation we shall refer to them as $w_1, w_2, \ldots, w_K$ and their outside vectors as $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_K$. In this question, you may not assume that the words are distinct. In other words, $w_i = w_j$ may be true when $i \neq j$ is true. Note that $o \notin \{w_1, \ldots, w_K\}$. For a center word c and an outside word o, the negative sampling loss function is given by:

$$\boldsymbol{J}_{\text{neg-sample}}(\boldsymbol{v}_c, o, \boldsymbol{U}) = -\log(\sigma(\boldsymbol{u}_o^\top \boldsymbol{v}_c)) - \sum_{k=1}^{K} \log(\sigma(-\boldsymbol{u}_k^\top \boldsymbol{v}_c)) \tag{6}$$

for a sample $w_1, \ldots w_K$, where $\sigma(\cdot)$ is the sigmoid function.

Compute the partial derivative of $\boldsymbol{J}_{\text{neg-sample}}$ with respect to a negative sample $\boldsymbol{u}_k$. Please write your answers in terms of the vectors $\boldsymbol{v}_c$ and $\boldsymbol{u}_k$, where $k \in [1, K]$. Hint: break up the sum in the loss function into two sums: a sum over all sampled words equal to u_k and a sum over all sampled words not equal to u_k.

$$J = -\log(\sigma(u_o^T v_c)) - \sum_{k=1}^{K} \log(\sigma(-u_k^T v_c))$$

k is no longer distinct

Now $\frac{\partial J}{\partial u_k}$ changes

$i = 1, 2, \ldots, K$

{ condition: $w_i = w_k$ }

Take derivative on $-\sum_{i=1 \mid w_i = w_k}^{K} \log(\sigma(-u_k^T v_c))$

Rui Wang's OH on Nook

Equivalently, $-\sum_{i \in \{1,\ldots K\}: w_i = w_k} \log(\sigma(-u_k^T v_c))$

Hence, $\frac{\partial J}{\partial u_k} = -\frac{\partial}{\partial u_k}\left[\sum_{i=1 \mid w_i = w_k}^{K} \log(\sigma(-u_i^T v_c))\right]$

$$= -\left[\sum_{i=1 \mid w_i = w_k}^{K} \frac{-v_c \sigma(-u_i^T v_c)(1 - \sigma(-u_i^T v_c))}{\sigma(-u_i^T v_c)}\right] \#$$

(h) (3 points) Suppose the center word is $c = w_t$ and the context window is $[w_{t-m}, \ldots, w_{t-1}, w_t, w_{t+1}, \ldots, w_{t+m}]$, where m is the context window size. Recall that for the skip-gram version of `word2vec`, the total loss for the context window is:

$$\boldsymbol{J}_{\text{skip-gram}}(\boldsymbol{v}_c, w_{t-m}, \ldots w_{t+m}, \boldsymbol{U}) = \sum_{\substack{-m \leq j \leq m \\ j \neq 0}} \boldsymbol{J}(\boldsymbol{v}_c, w_{t+j}, \boldsymbol{U}) \tag{7}$$

Here, $\boldsymbol{J}(\boldsymbol{v}_c, w_{t+j}, \boldsymbol{U})$ represents an arbitrary loss term for the center word $c = w_t$ and outside word w_{t+j}. $\boldsymbol{J}(\boldsymbol{v}_c, w_{t+j}, \boldsymbol{U})$ could be $\boldsymbol{J}_{\text{naive-softmax}}(\boldsymbol{v}_c, w_{t+j}, \boldsymbol{U})$ or $\boldsymbol{J}_{\text{neg-sample}}(\boldsymbol{v}_c, w_{t+j}, \boldsymbol{U})$, depending on your implementation.

Write down three partial derivatives:

(i) $\partial \boldsymbol{J}_{\text{skip-gram}}(\boldsymbol{v}_c, w_{t-m}, \ldots w_{t+m}, \boldsymbol{U}) / \partial \boldsymbol{U}$

(ii) $\partial \boldsymbol{J}_{\text{skip-gram}}(\boldsymbol{v}_c, w_{t-m}, \ldots w_{t+m}, \boldsymbol{U}) / \partial \boldsymbol{v}_c$

(iii) $\partial \boldsymbol{J}_{\text{skip-gram}}(\boldsymbol{v}_c, w_{t-m}, \ldots w_{t+m}, \boldsymbol{U}) / \partial \boldsymbol{v}_w$ when $w \neq c$

Write your answers in terms of $\partial \boldsymbol{J}(\boldsymbol{v}_c, w_{t+j}, \boldsymbol{U}) / \partial \boldsymbol{U}$ and $\partial \boldsymbol{J}(\boldsymbol{v}_c, w_{t+j}, \boldsymbol{U}) / \partial \boldsymbol{v}_c$. This is very simple – each solution should be one line.

$$\partial J_{\text{skip-gram}} / \partial U = \sum_{\substack{-m \leq j \leq m \\ j \neq 0}} \frac{J(v_c, w_{t+j}, U)}{\partial U}$$

$$\partial J_{\text{skip-gram}} / \partial U = \sum_{\substack{-m \leq j \leq m \\ j \neq 0}} \frac{J(v_c, w_{t+j}, U)}{\partial v_c}$$

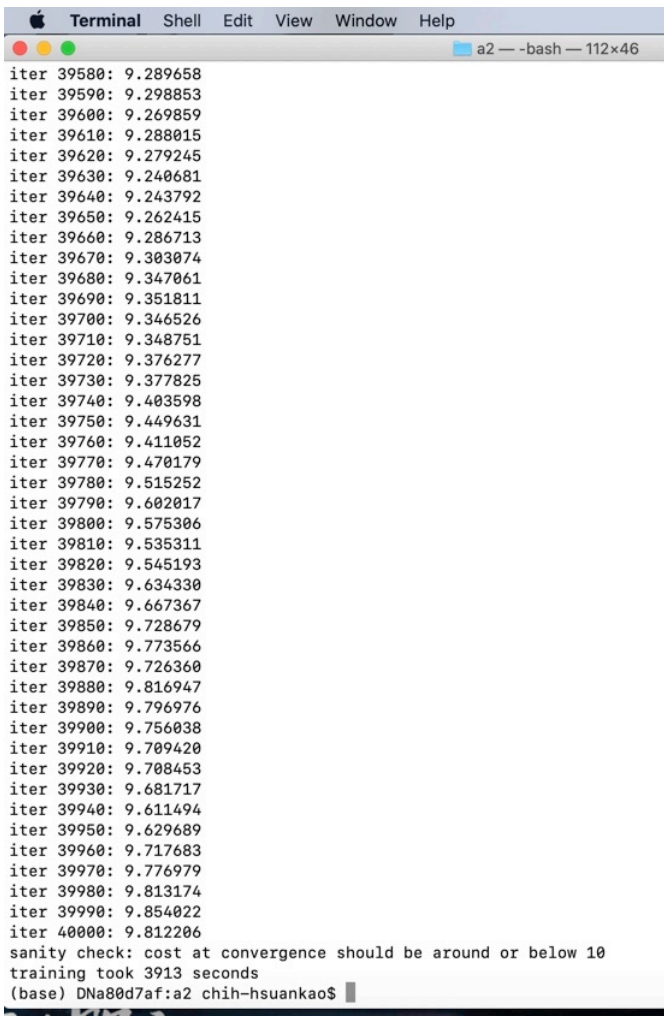
$$\partial J_{\text{skip-gram}} / \partial w_c = 0 \qquad w \neq c$$

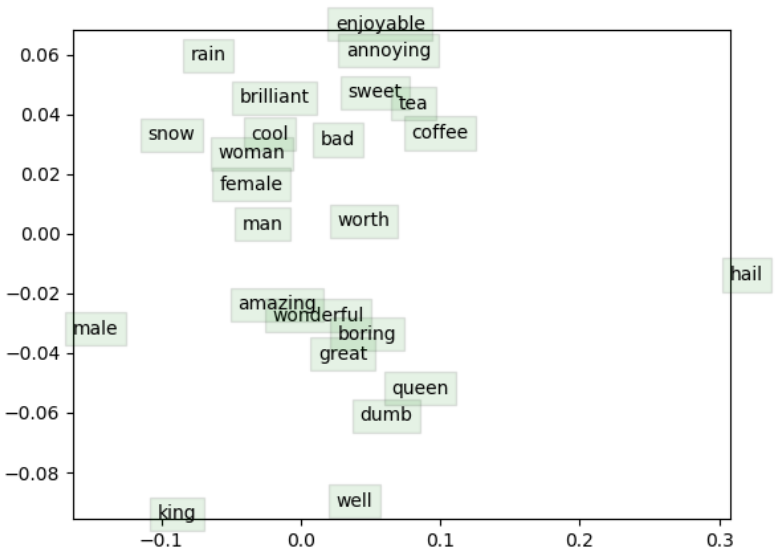
2 Coding: Implementing word2vec

(c) (2 points) Show time! Now we are going to load some real data and train word vectors with everything you just implemented! We are going to use the Stanford Sentiment Treebank (SST) dataset to train word vectors, and later apply them to a simple sentiment analysis task. You will need to fetch the datasets first. To do this, run `sh get_datasets.sh`. There is no additional code to write for this part; just run `python run.py`.

Note: The training process may take a long time depending on the efficiency of your implementation and the compute power of your machine ***(an efficient implementation takes one to two hours)****. Plan accordingly!*

After 40,000 iterations, the script will finish and a visualization for your word vectors will appear. It will also be saved as `word_vectors.png` in your project directory. **Include the plot in your homework write up.** Briefly explain in at most three sentences what you see in the plot.





I roughly used different colors to create clusters that I deem proper — And from this, there're indeed good clusters such as [amazing, wonderful], [woman, female], [sweet, tea, coffee], etc. However, an innegligible portion of the word embedding doesn't seem to be that ideal, for example, 'boring' clusters with positive words. Also, 'hail' is out of frame due to word2vec's inability to handle out-of-vocabulary words, as word2vec algo is based on the contextual window structure.

Observation: