VU Thi Hai Yen
haiyen96.hp@gmail.com

1 Question 1

We have

$$L(t, \mathcal{C}_t^+, \mathcal{C}_t^-) = \sum_{c \in \mathcal{C}_t^+} \log(1 + e^{-w_c \cdot w_t}) + \sum_{c \in \mathcal{C}_t^-} \log(1 + e^{w_c \cdot w_t}). \tag{1}$$

Thus,

$$\begin{aligned} \frac{\partial L}{\partial w_{c^+}} &= \frac{\partial \log(1 + e^{-w_{c^+} \cdot w_t})}{\partial w_{c^+}} \\ &= \frac{1}{1 + e^{-w_{c^+} \cdot w_t}} \cdot \frac{\partial e^{-w_{c^+} \cdot w_t}}{\partial w_{c^+}} \\ &= \frac{e^{-w_{c^+} \cdot w_t}}{1 + e^{-w_{c^+} \cdot w_t}} \cdot \frac{\partial(-w_{c^+} \cdot w_t)}{\partial w_{c^+}} \\ &= -w_t(1 - \sigma(w_{c^+} \cdot w_t)), \end{aligned} \tag{2}$$

where $\sigma(x) = \frac{1}{1 + e^{-x}}$. Similarly, we obtain that

$$\frac{\partial L}{\partial w_{c^-}} = \frac{\partial \log(1 + e^{w_{c^-} \cdot w_t})}{\partial w_{c^-}} = w_t(1 - \sigma(-w_{c^-} \cdot w_t)). \tag{3}$$

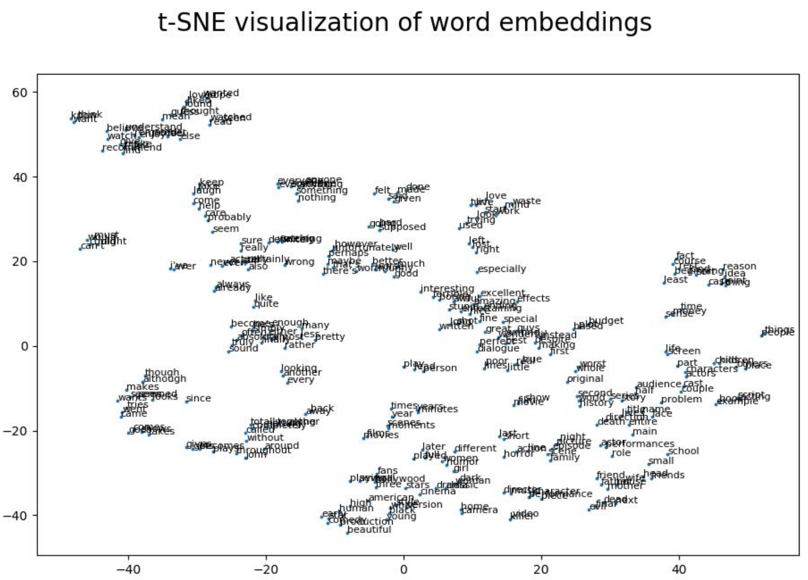
2 Question 2

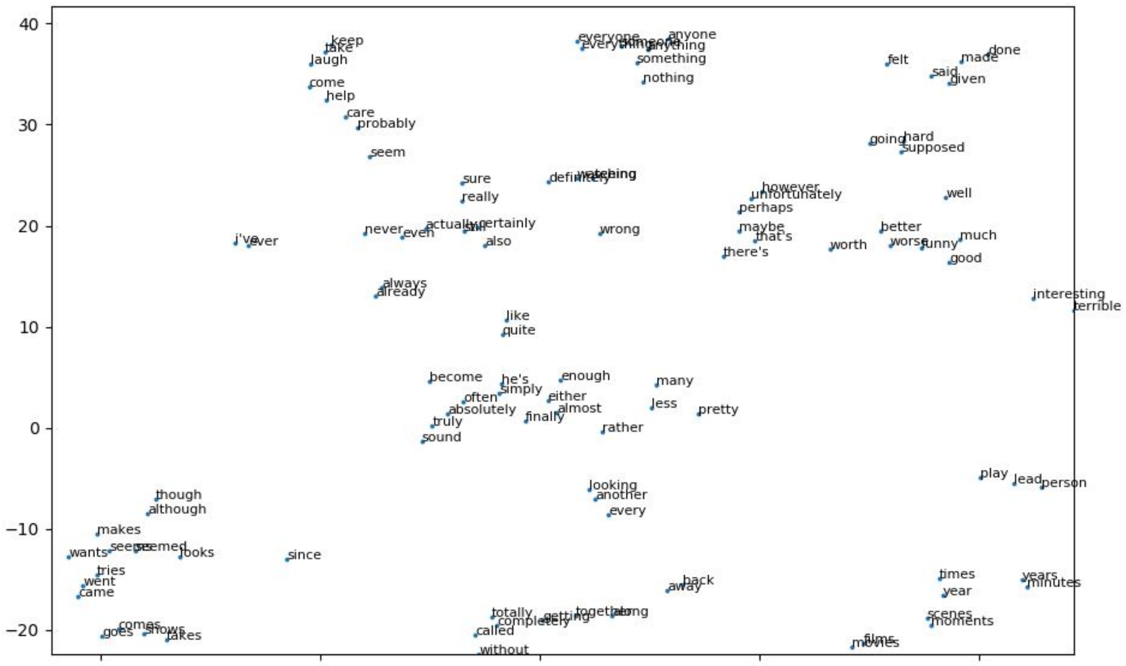
From equation (1), we derive

$$\begin{aligned} \frac{\partial L}{\partial w_t} &= \sum_{c \in \mathcal{C}_t^+} \frac{\partial \log(1 + e^{-w_c \cdot w_t})}{\partial w_t} + \sum_{c \in \mathcal{C}_t^-} \frac{\partial \log(1 + e^{w_c \cdot w_t})}{\partial w_t} \\ &= \sum_{c \in \mathcal{C}_t^+} -w_c(1 - \sigma(w_c \cdot w_t)) + \sum_{c \in \mathcal{C}_t^-} w_c(1 - \sigma(-w_c \cdot w_t)) \end{aligned} \tag{4}$$

3 Question 3

We obtain the following t-SNE visualization of the embedding space:





We see that the space spreads into different clusters. If we take a zoom at the figure: We see that the words close to each other are indeed similar in some meanings. For example, the synonym words **films** and **movies** are very close to each other, the words **times**, **years** and **minutes**, which refer to the time, are close as well. We also observe an interesting analogy behavior of the embedding space, the present form of verbs **wants**,**tries**,**takes**,etc. lies close to each other (down-left corner) while the past participle form of the verbs like **felt**,**made**, **given**, etc. are found at the up-right corner, and the non conjugated verbs like **laugh**, **come**, **help** lies in the top-left of the image.
The following table shows the cosine similarity between some chosen words. We see that the embedding vectors are highly reasonable.

	banana	watch	movie
bread	0.28	0.20	0.17
listen	0.08	0.92	0.89
film	-0.03	0.92	0.99

4 Question 4

We base our solution on [1]. To learn the document vectors, we create another embedding M_D for the documents which lives in $\mathbb{R}^{|D|\times m}$, where m is the embedding size of the document vectors.
In the preprocessing step, each document d is assigned to an unique integer i_d in $1, \dots, |D|$, then the i_d-th row of the embedding matrix M_D represents the vector m_d of that document.
To train the vectors, we can either concatenate/average the document vector m_d to the vectors of context words w_c in the document in order to predict the target word w_t, as in the CBOW model, or we can use the document vector m_d as an input to predict the words w_d in the document.
The first method, which is called Distributed Memory Model of Paragraph Vectors (PV-DM), is trained using the following log-likelihood:

$$\arg\max_{\theta} \sum_{d\in D}\sum_{t\in d} \log p(t|d, \mathcal{C}_t;\theta), \tag{5}$$

where $\mathcal{C}_t$ is the context words of t.
The second method, which is called Distributed Bags of Words version of Paragraph Vector (PV-DBOW), is trained using the following log-likelihood:

$$\arg\max_{\theta} \sum_{d\in D}\sum_{t\in d} \log p(t|d;\theta). \tag{6}$$

Based on the skip-gram model that we have implemented, we suggest a third method to train the document vector which use a concatenated/averaged vector of the document and the target word to predict the context word, which utilises the following log-likelihood:

$$\arg\max_{\theta} \sum_{d \in D} \sum_{t \in d} \sum_{c \in \mathcal{C}_t} \log p(c|d, t; \theta). \tag{7}$$

We can use either use one of the three methods to train the document embedding, or using all the methods and concatenate the vectors to obtain the final vector of the document.
If we want to change our model to use the third method, we can simply choose $m = d$, and we can simply take the average vector of the target word and the document. In this case, our loss function is determined as:

$$\arg\min_{\theta} \sum_{d \in D} \sum_{t \in d} \left(\sum_{c \in \mathcal{C}_t^+} \log(1 + e^{-w_c \cdot (w_t + w_d)/2}) + \sum_{c \in \mathcal{C}_t^-} \log(1 + e^{w_c \cdot (w_t + w_d)/2}) \right). \tag{8}$$

And we can compute the gradients with respect to W_c, W_t and W_d as in previous questions.

References

[1] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on International Conference on Machine Learning*, ICML'14, pages 1188–1196, 2014.