# 1 Question 1

Let us consider our loss for a given training example $(t, C_t^+, C_t^-)$:

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

Now we calculate the partial derivative of the loss w.r.t one positive example.

$$\frac{\partial L}{\partial w_{c^+}} = \frac{\partial}{\partial w_{c^+}}[\sum_{c \in C_t^+} \log(1 + e^{-w_c \cdot w_t})] + \frac{\partial}{\partial w_{c^+}}[\sum_{c \in C_t^-} \log(1 + e^{+w_c \cdot w_t})] \tag{2}$$

We can see that $w_{c^+}$ is in a single term of the first sum. The rest of the terms of this first sum and the second sum don't depend of $w_{c^+}$ so their derivative is zero. Using the chain rule we have now:

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

This is a vector with the same dimension of $w_t$. Now we do the same to calculate the partial derivative of the loss w.r.t one negative example.

$$\frac{\partial L}{\partial w_{c^-}} = \frac{1}{1 + e^{w_{c^-} \cdot w_t}} * (w_t * e^{w_{c^-} \cdot w_t}) = \frac{\mathbf{w_t}}{\mathbf{1 + e^{-w_{c^-} \cdot w_t}}} \tag{4}$$

This is also a vector with the same dimension of $w_t$.

# 2 Question 2

Considering the same loss function of the previous question, now Compute the partial derivative of the loss w.r.t. the target word $w_t$. This time we calculate the derivative in both sums.

$$\frac{\partial L}{\partial w_t} = \frac{\partial}{\partial w_t}[\sum_{c \in C_t^+} \log(1 + e^{-w_c \cdot w_t})] + \frac{\partial}{\partial w_t}[\sum_{c \in C_t^-} \log(1 + e^{+w_c \cdot w_t})] \tag{5}$$

$$\frac{\partial L}{\partial w_t} = \sum_{c \in C_t^+} \frac{\partial}{\partial w_t}[\log(1 + e^{-w_c \cdot w_t})] + \sum_{c \in C_t^-} \frac{\partial}{\partial w_t}[\log(1 + e^{+w_c \cdot w_t})] \tag{6}$$

Using the chain rule:

$$\frac{\partial L}{\partial w_t} = \sum_{\mathbf{c} \in \mathbf{C_t^+}} \frac{-\mathbf{w_c}}{\mathbf{1 + e^{w_c \cdot w_t}}} + \sum_{\mathbf{c} \in \mathbf{C_t^-}} \frac{\mathbf{w_c}}{\mathbf{1 + e^{-w_c \cdot w_t}}} \tag{7}$$

# 3 Question 3

Regarding the results of the cosinus similarity of the words I obtained the following results:

- movie-film = 0.9949
- movie-banana = 0.0192
- film-banana = -0.0006

This makes a lot of sense since movie and film share more semantic meaning than banana. This means that our word embedding representation of the words is correct since it manages to take into account the semantics of the words.

With respect to embedding space, we note that words that have proximate or somehow related meanings are closer together than those that do not share these characteristics. As we can see in Fig. 1, the words

”felt”, ”feel”, ”loved” are together since they share similar meanings, we also have the case of ”waste” and ”worse” that although they do not share the same meaning, they both have a negative connotation. We also see that words that do not seem to have a relationship are very distant, this is the case of ”long” and ”death”. In general we see small clusters that although not all possess a relationship of the same meaning, they share some relationship that allows them to be close.

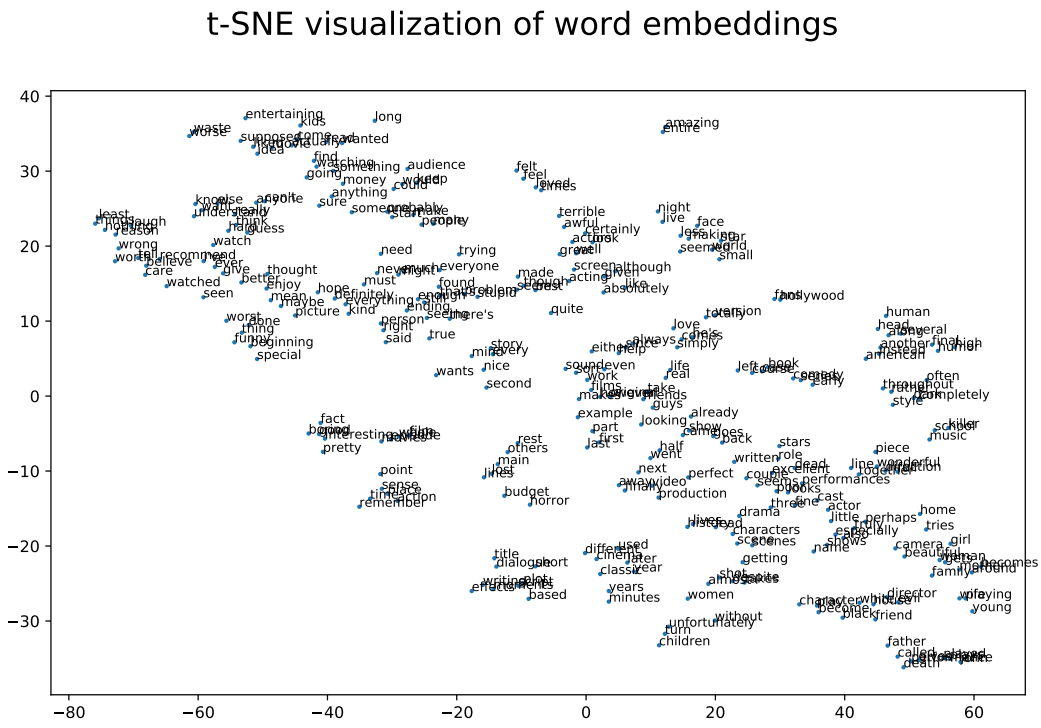


Figure 1: t-SNE visualization of each word according to its word embedding vector in 2D dimension.

# 4 Question 4

According to [1], there are two ways to do this. The first is by using the distributed memory model (PV-DM) in which we use Wd, which represents the vectors of the document, and Wt to predict the context. According to the article, Wd would represent in this case a kind of memory that contains the missing information in the context. This is an expensive method since we compare word vectors with word vectors.

The other way to do it is to follow the idea of the skip gram using the other method detailed in the article, called Paragraph Vector with distributed bag of words (PV-DBOW) in which Wt is replaced by Wd, and with it we try to predict the context. This method is less expensive in terms of storage and faster to execute than the previous one, given that word vectors are only compared to document vectors.

However, for best results the author indicates that it is best to use a concatenation of both results to represent the document vectors.

Regarding the preprocessing we need to indicate which document each window created belongs to, this is done by modifying the function sample_examples().

# References

[1] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *International conference on machine learning*, page 1188–1196, 2014.