Virgile Rennard
virgile@rennard.org

Lab session # 2

ALTEGRAD 2019

12/19/19

1 Question 1

First off, let's compute the derivative $wrt\ \partial\omega_{c^+}$

$$\begin{aligned}\frac{\partial L(t, C_t^+, C_t^-)}{\partial\omega_{c^+}} &= \frac{\partial}{\partial\omega_{c^+}}(\sum_{c\in C_t^+} log(1+e^{-\omega_c\omega_t}) + \sum_{c\in C_t^-} log(1+e^{\omega_c\omega_t})) \\ &= \frac{-\omega_t e^{-\omega_t\omega_c}}{1+e^{-\omega_t\omega_c}} \\ &= \frac{-\omega_t}{1+e^{\omega_t\omega_c}}\end{aligned} \tag{1}$$

and let's compute the derivative $wrt\ \partial\omega_{c^-}$

$$\begin{aligned}\frac{\partial L(t, C_t^+, C_t^-)}{\partial\omega_{c^-}} &= \frac{\partial}{\partial\omega_{c^-}}(\sum_{c\in C_t^+} log(1+e^{-\omega_c\omega_t}) + \sum_{c\in C_t^-} log(1+e^{\omega_c\omega_t})) \\ &= \frac{\omega_t e^{\omega_t\omega_c}}{1+e^{\omega_t\omega_c}} \\ &= \frac{\omega_t}{1+e^{-\omega_t\omega_c}}\end{aligned} \tag{2}$$

2 Question 2

let's compute the derivative $wrt\ \partial\omega_t$

$$\begin{aligned}\frac{\partial L(t, C_t^+, C_t^-)}{\partial\omega_t} &= \frac{\partial}{\partial\omega_t}(\sum_{c\in C_t^+} log(1+e^{-\omega_c\omega_t}) + \sum_{c\in C_t^-} log(1+e^{\omega_c\omega_t})) \\ &= \sum_{c\in C_t^+} \frac{-\omega_c e^{-\omega_c\omega_t}}{1+e^{-\omega_c\omega_t}} + \sum_{c\in C_t^-} \frac{\omega_c e^{\omega_c\omega_t}}{1+e^{\omega_c\omega_t}} \\ &= \sum_{c\in C_t^+} \frac{-\omega_c}{1+e^{\omega_c\omega_t}} + \sum_{c\in C_t^-} \frac{\omega_c}{1+e^{-\omega_c\omega_t}}\end{aligned} \tag{3}$$

3 Question 3

3.0.1 Interpreting the similarity values

The cosine similarity is a function which measures how similar two vectors are by measuring the angle between them. Therefore, if two vectors happen to have the same orientation, we will have a similarity of 1, and to with a 90 angle will have a similarity of 0, while opposite vectors will have one of -1.
The idea is that two words with a similar meaning will give us a cosine similarity close to one, as we can see with "men" and "women" having a similarity close to 1, while "boots" and "banana" which aren't similar get us a small one

3.0.2 T-Sne and embedding space

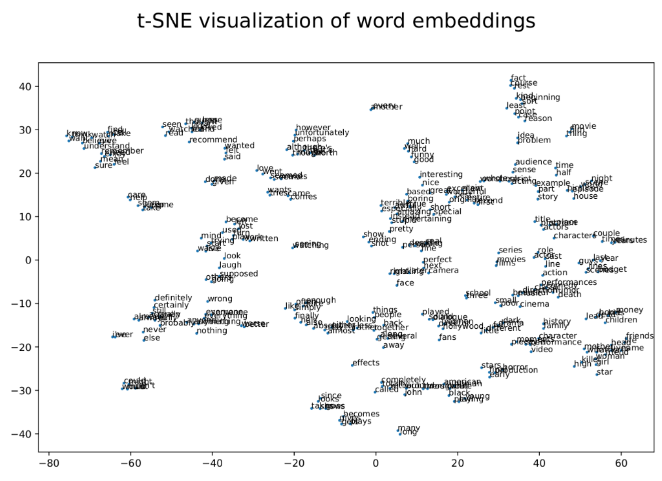

Figure 1: Comparison of our different methods scores

T-Sne is used to visualize similarity between objects located in a high dimensional space. With it, we can observe that skip-gram is a great method for learning word embeddings and capture semantic relationships. Similar words tend to end up close to one another.

4 Question 4

We base our answer on the following paper [1].

In this paper is presented a way to find embeddings not only for words, but for whole paragraphs or documents at the same time, while still keeping semantically similar embeddings, and keeping the position of words in whole paragraphs. While other approaches have been used to get paragraph vectors, such as [2], only paragraph vector can both keep the order of the words into account as well as working on more than sentences. One of its main interest is that, unlike other methods such as Recursive Neural Tensor Networks, it requires *no parsing* and no specific tuning of word weighting functions
Every paragraph will be mapped to a vector in a matrix D to get their embeddings, while as usual, words will be mapped to a vector in a matrix W, they are concatenated to predict the next word in a context, trained during the prediction task. Concatenate the paragraph vector with several word vectors from a paragraph to predict the following word in the given context. Changing our estimation in the following manner :

$$y = b + Uh(\omega_{t-k}, \dots, \omega_{t+k}; W, D) \tag{4}$$

Which depends not only on the word vectors as in the word embedding skip grams, but also on the paragraph vectors, updating our Paragraph vector parameters during backpropagation. The paragraphs are in fact just another ”word” working on the meaning of the paragraph.
Here, the context is of fixed length, sampled from a window on the paragraph. The paragraph vector is shared across the contexts generated from the paragraph, but not across all paragraphs, unlike the word embeddings which are shared throughout the documents
The Paragraph embeddings as well as word embeddings are trained using *SGD* and backpropagation. Just as the word embeddings are estimated during the prediction task, the paragraph embeddings will be estimated this way. However, we have to perform an inference step where we freeze the other parameters (W embeddings and the softmax U,b)
Another method is presented where you instead ignore the econtext words in the input and force the model to predict words randomly sampled from the paragraph in the output. What changes is taht at each iteration of SGD, we take a random text window and perform a classification task with the paragraph vector

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 - Volume 32*, ICML’14, pages II–1188–II–1196. JMLR.org, 2014.

[2] Jeff Mitchell and Mirella Lapata. Composition in distributional models of semantics. *Cognitive Science*, 34(8):1388–1429, 2010.