Firas
Omrane

Lab session # 2
ALTEGRAD 2019

10/24/19

1 Question 1

This is my answer to question 1. Below is an equation:

$$\frac{dL}{dw_{c_+}} = \frac{-w_t.exp(-w_{c_+}w_t)}{1+exp(-w_{c_+}w_t)} = \frac{-w_t}{1+exp(w_{c_+}w_t)}$$

$$\frac{dL}{dw_{c_-}} = \frac{w_t}{1+exp(-w_{c_-}w_t)}$$

2 Question 2

$$\frac{dL}{dw_t} = \sum_{c\in C_t^+} \frac{-w_c}{1+exp(w_c w_t)} + \sum_{c\in C_t^-} \frac{w_c}{1+exp(-w_c w_t)}$$

3 Question 3

We observe that we have build the word embedding for each word in our vocabulary with respect to the (IMDB) movie database. So now we have a more semantic representation of our words, for instance, related words have close embedding (see Figure 1). Like (movie,film) and (bad,good).

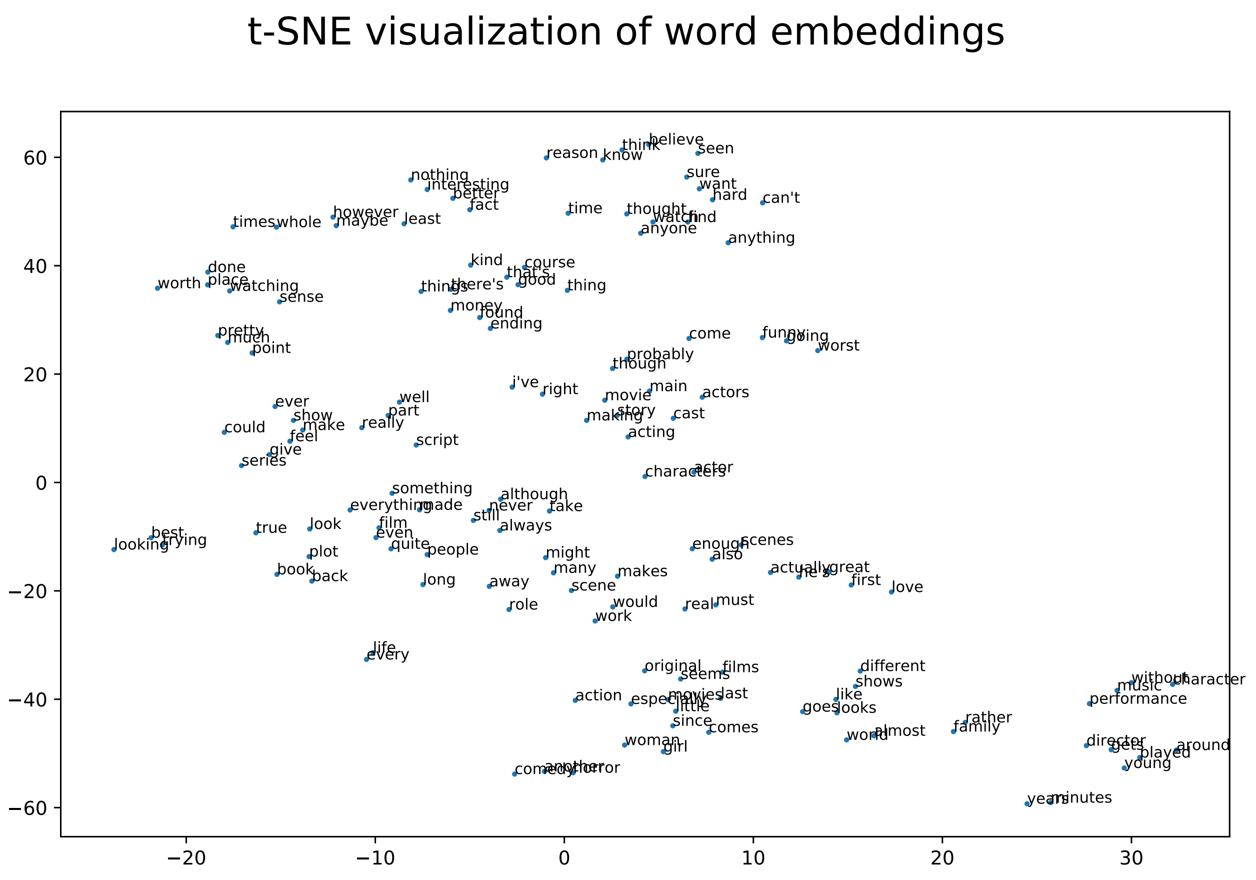

Figure 1: Word embedding on IMDB.

cosine similarity between movie and film is : 0.9962
cosine similarity between bad and good is : 0.992
cosine similarity between funny and humor is : 0.9855
cosine similarity between good and awesome is : 0.9179

And the words that are unrelated have embedding that are far.
cosine similarity between movie and banana is : 0.1081
cosine similarity between women and olds is : -0.0966

But still not trained with large corpus so if we take the embeddings of word2vec that are trained on wikipedia corpus and then fine tune it with our data we will get more relaible results.

4 Question 4

There is one solution that do not need any changes in the pipeline which consists of just doing a weighted averaging of word vectors, but this loses the word order in the same way as the standard bag of words models do.

But a more accurate method would be just like stated in [1] where we can learn another matrix D such that every document is mapped to a unique vector. The dimension of D would be $N * p$ where N is the number of the documents in our corpus and p is the dimension of our learnt documents vectors. The document vector is shared across all contexts generated from the same document. The word vector matrix W, however, is shared across documents.

The idea will be a combination of the two methods presented in [1]: the Distributed Memory Model of Paragraph Vectors (PV-DM), where CBOW model is used, and the Distributed Bag of Words version of Paragraph Vector (PV-DBOW) that used a skip-gram like method to learn only the document vectors.

At training time we will average/concatenate the word vector and the corresponding document vector that will serve like 'the topic or context' for the word and we try to predict the other window words just like skip-gram and then we update both the parameters of the matrix W and the used vector from the matrix D.

About the changes in the pipeline we have to:
- Precise for each window produced the corresponding document and this can be done by just adding an index in the windows list indicating for each window tuple the conrresponding document.

- In the training at each step we have to calculate the gradients and update the weight vectors. The equations of the gradients $\frac{dL}{dw_{c_+}}$, $\frac{dL}{dw_{c_-}}$ and $\frac{dL}{dw_t}$ will change and will include the document vector. And we have also to calculate $\frac{dL}{dw_d}$ where w_d is the vector of the document at each step.

References

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