CLASS Exercises: WORD2VEC

Exercise 1

A feed-forward neural network language model (LM) is an alternative architecture for training word vectors. This architecture focuses on predicting a word given the N previous words. This is done by concatenating the word vectors of N previous words and use them as input of a single hidden layer of size H with a non-linearity (e.g. tanh). Finally, a softmax layer is used to make a prediction of the current word. The size of the vocabulary is V. The model is trained using a cross entropy loss for the current word.

Let the word vectors of the N previous words be $x_1; x_2; \ldots x_N$, each a column vector of dimension D, and let $y$ be the one-hot vector for the current word. The network is specified by the equations that follow these lines:

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}$$

$$h = \tanh(Wx + b)$$

$$\hat{y} = \text{softmax}(Uh + d)$$

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

$$CE = -\sum_i y_i \log(\hat{y}_i)$$

The dimensions of our parameters and variables are $x \in R^{(N \cdot D)}, W \in R^{H \times (N \cdot D)}, b \in R^H, h \in R^H, U \in R^{V \times H}, d \in R^V, \hat{y} \in R^V$.
4a. Mention 2 important differences between this feed-forward neural network LM and the CBOW model. Explain how these differences might affect the word vectors obtained.

4b. Compute the complexity of forward propagation in a feed-forward LM for a single training example. Propose at least one way to change the model that would reduce this complexity.

**Exercise 2.**

5a. We know that dense word vectors like the ones obtained with word2vec or GloVe have many advantages over using sparse one-hot word vectors. Name a few.

5b. Also name at least 2 disadvantages of sparse vectors that it are not solved in dense vectors. Which of the following is NOT an advantage dense vectors have over sparse vectors?

**Exercise 3**

Given the following neural architecture. What is it learning? Can you explain which exact NLP task is training?

![Neural Architecture Diagram]

**Exercise 4**

We have each used the Word2Vec algorithm to obtain word embeddings for the same vocabulary of words \( V \).

In particular, developer A has got `context' vectors \( u_w^A \) and `center' vectors \( v_w^A \) for every \( w \) in \( V \), and developer B has got `context' vectors \( u_w^B \) and `center' vectors \( v_w^B \) for every \( w \) in \( V \).

For every pair of words \( w, w' \in V \), the inner product is the same in both models: \( \langle u_w^A, v_{w'}^A \rangle = \langle u_w^B, v_{w'}^B \rangle \). Does it mean that, for every word \( w \) in \( V \), \( v_w^A = v_w^B \)? Discuss your response.