Master MVA - Introduction to Deep Learning
NLP Deliverable - 05/03/2016
Achari Berrada Youssef

Part I: Word and Sentence embeddings with Word2vec:

Question 1: First Part

1. The total number of raw words find in the corpus is 85026035.
2. The number of words retained in the word2vec vocabulary is 71290 (with default min_count = 5).

Question 2: Second Part

1. Similarity between "apple" and "mac" : 0.567861632452
   Similarity between "apple" and "peach" : 0.178399832237
   Similarity between "banana" and "peach" : 0.688715470006

   We notice that, even though the relation that links "apple" and "peach" is the same as the relation that links "banana" and "peach", these two pairs of words have not the same similarity measure. And it could be explained by the first pair ("apple","mac"), in the sense that the word "apple" has moved to "mac" and therefore didn't keep a similarity with *"the fruits"*.
2. The closest word to "difficult":
   - for model : "easy"
   - for model_phrase : "very_difficult"

   In model, the most similar word is the contrary word. In model_phrase, it is a phrase with two words that has the same meaning (exaggerated).
   Three phrases closest to the word **"Clinton"** are:
   'bush', 0.8667015433311462;
   'reagan', 0.859365701675415;
   'gore', 0.8575333952903748.
3. The closest word to the vector (**France - Germany + Berlin**) is **Paris** with 0.757699728012085 similarity measure.
4. We can find some similarity between words, like what is a meal for a country based on a meal I know in my country:
   Pizza for Italian is sauce for French with score 0.723899781704
   Couscous for Moroccan is champagne for French with score 0.567590713501
   Here is another example for cars:
   Ferrari for Italian is benz for Germans with score 0.66373950243

Question 3: Third part.

1. The sentence 777 is "a girl trying to get fit for a bodybuilding competition ."
   The closest sentence to it is "gymnasts get ready for a competition ." with a similarity score of 0.902949750423.
2. Sentence 1 : id 8827 -> gymnasts get ready for a competition .

   score: 0.902949750423
   Sentence 2 : id 1476 -> a woman is getting ready to perform a song for the audience .

score: 0.89009732008
Sentence 3 : id 6232 -> a runner in a competition want to go to the finish line .

score: 0.855536520481
Sentence 4 : id 5771 -> men working to build a fence for customers .

score: 0.851471722126
Sentence 5 : id 5684 -> a man prepares to give a speech for a television audience .

score: 0.849476158619

Question 4: Fourth part.

1. IDF score of the words :
   'a': 0.68277890779
   'the': 1.25191644135
   'clinton': The word clinton does not appear in the sentences. So it has the maximum idf ($+\infty$ or $\log(Number\_of\_docs)$ if we consider the smooth idf).

2. The closest sentence with the IDF weights to the query sentence with idx = 777: 'a girl trying to get fit for a bodybuilding competition .' is 'gymnasts get ready for a competition .' with a score of : 0.897237718105

Part II: Simple LSTM for Sequence Classification:

Question 1:

1. The input of the embedding layer is in minibatch: (32×80), where 32 is the batch_size and 80 is the input_lenght[1].
   The partition of the input data is :

Training set	Validation set	Test set
28000	7000	15000

2. The output of the embedding layer is 32.

3. The output of the LSTM layer is 64.

Question 2:

1. The number of parameters is $504,897$.
   The number of sentences in the training set is $28,000$ (without counting the validation set).
   In order to train a model that has many parameters compared to the number of samples, we use **Dropout**.

Question 3:

1. The T states of the LSTM are fed into a Dense layer which is just in Keras a regular fully connected NN layer, with bias. Then the output is fed into a sigmoid activation function. Therefore, we have:

$$f(h_1, ..., h_T) = \sigma\left(\sum_{i=1}^{T} w_i * h_i + b\right) \qquad \text{where } \sigma \text{ is the sigmoid function.}$$

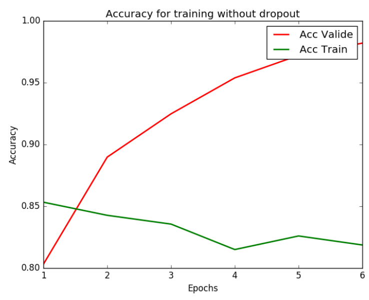
Question 4:

1. Plot of the accuracy w.r.t. epochs: (see figure 1).

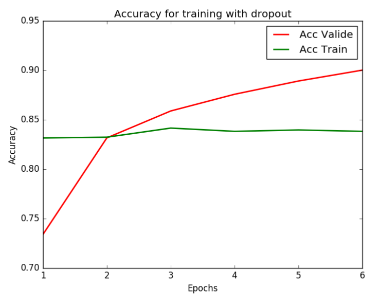
Question 5:

1. In the Stochastic Gradient Descent (SGD) optimization method, we update the parameters with gradient descent where we approximate the true gradient by the gradient at a single example at each iteration step. The Adaptative Moment Estimation (Adam) is a method that update both the gradients like in the momentum and the magnitude like RMSprop at each iteration of the gradient descent method.

[1] The input of embedding layer is in the shape of (input_dim, input_lenght) as explained in the keras documentation website: https://keras.io/layers/embeddings/



(a) Without Dropout: Accuracy on test set = 0.8212



(b) With Dropout: Accuracy on test set = 0.8390

Figure 1: plots of the Accuracy on Training and Validation set

Part III: Simple ConvNet For Sequence Classification:

Question 1:
Test score: 0.370939202944
Test accuracy: 0.837466666635

Question 2: For the Convolution1D, we have :

- Input shape is $S \times E$ with S is the number of words and E the dimension of the word embedding. Here we have $S = 5000$ and $E = 16$.
- Output shape: