MASTER OF SCIENCE IN ENGINEERING

Teachers: J. Hennebert, M. Melchior
Assistants: J. Esseiva, C. Gisler, C. Huwyler

Practical work 11 – 6/12/2018
Recurrent Neural Networks with Keras - part 2

Objectives

The objective of this PW is to practice with some applications of Recurrent Neural Networks (RNN) and with word embeddings.

Submission

— **Deadline** : Wednesday, 19 December, 12am

— **Format** : Zip with report and iPython notebooks.

Exercice 1 Human Activity Recognition cont'd

Continue the exercise of PW11 on Human Activity Recognition.

Try with different layers of LSTMs and/or GRUs to achieve an accuracy as good as you can.

Exercice 2 Sentiment Analysis

In this exercise you should create and study different models for sentiment analysis in *keras*. This time we do not provide an iPython notebook. Rather, you should structure it yourself as part of the exercise.

Conduct the following steps :

a) **Load Data** : Load the training and test data from `emoji_data/train.csv` and `emoji_data/test.csv` in the zipped `emoji_data.zip`).
The labels correspond to the emojis which can be loaded as specified below :

```
1
2 import emoji
3 emoji_dictionary = {"0": "\u2764\uFE0F",    # :heart: is sometimes in black
4                     "1": ":baseball:",
5                     "2": ":smile:",
6                     "3": ":disappointed:",
7                     "4": ":fork_and_knife:"}
8
9 def label_to_emoji(label):
10     return emoji.emojize(emoji_dictionary[str(label)], use_aliases=True)
```

Here, you need to install the *emoji* package (e.g. by `pip install emoji`).

b) **Define Embedding** : The dataset is very small so that the word embedding cannot be learned. Rather you should use a pre-trained word embedding. Here, you should first use the glove embedding (*glove*.6*B*.50*d.txt*, see https ://nlp.stanford.edu/projects/glove/).
In order to make the pre-trained embedding work you should load this file into a dict which maps word to vectors. Then you should also prepare a dict which maps words to indices. This will allow in the next step to transform sentences of the the emoji dataset (sequences of words) to sequences of integers.
Furthermore, you need to prepare a weights matrix that will provide the mapping from the indices to vectors.
Finally, instantiate an Embedding layer (`keras.layers`). Consult the resources on the Internet and Keras documentation.

c) **Input Preparation** : Transform the sentences of the emoji input data to sequences of integers. Use padding to make all the sentences equal in length (use e.g. the *sequence* class in *keras.preprocessing*). Furthermore, transform the labels into one-hot encoded vectors.

d) **Model A** : As a first model, just average the output of the embedding (along the sequence direction) and classify that with a softmax. Make sure that you only average the non-zero entries of the sequence. Define, compile and train the model.

Check the accuracy you can achieve.

List all the wrongly classified test sentences and try to give an interpretation where the classifier has problems. Also check the confusion matrix.

e) **Model B** : Now, use two LSTM layers after the embedding layer and feed the output again to a softmax. Can you achieve improved performance ?

Again list all the wrongly classified test sentences and try to give an interpretation where the classifier has problems.

f) **Comparison** Compare the performance of models A and B. Is there a benefit of using LSTMs ? Explain.

g) **Other Pre-Trained Embeddings** : Now try another embedding (e.g. $glove.6B.100d.txt$ and/or word2vec from gensim) and describe and interpret what you obtain.

h) **OPTIONAL – Other datasets** : Apply the same analysis to other datasets such as the IMDB dataset or the Twitter Airline dataset (see e.g. https://www.kaggle.com/crowdflower/twitter-airline-sentiment) or even your own dataset. Try different models and regularisation scheme to achieve the best accuracy you can.

Exercice 3 Optional : Review Questions

a) In what situations would you expect LSTMs (by its design) to perform better than SimpleRNNs ?

b) Describe why SimpleRNNs have problems in learning long-term dependencies.

c) Compute the number of parameters to be trained for a two-layer $LSTM$ and $softmax$ with hidden state dimensions 32 and 64, respectively, 10 classes to classify in the softmax and inputs given by sequences of length 100 and each element a vector of dimension 30.

d) What is the difference between the word2vec embedding and the embedding learned on the IMDB corpus for positive/negative sentiment analysis.