Teachers: J. Hennebert, M. Melchior
Assistants: Y.-I. Beffa, S. Hackstein, O. Zayene

Practical Work 11 – 05/05/2022
Generative RNN - Word Embedding

Objectives

The objective of this PW is to experiment with generative RNNs and word embeddings. You can do the exercise using your favorite framework, either with Tensorflow or Pytorch.

Submission

— **Deadline** : Wednesday, 18th of May, 12am (noon)

— **Format** : Zip with your jupyter notebooks.

Exercise 1 Sequence generation - startup names

The objective of this exercise is to build a generator of startup names using a RNN. We will train the system with a corpus of existing startup names composed of 172K entries. Figure 1 illustrates the 20 first entries of the corpus.

Many-to-one approach

a) Read again the examples on the Shakespeare text generation in the section *Generative RNN* from the class support. Make sure you understand the different steps for the data preparation. You can also start from the ipython notebooks provided on Moodle for the Shakespeare text generation.

b) Get from Moodle the file `companies.csv` that will be used to train the system.

c) Read the data from the file. Treat the whole set of names as a unique string of text, like in the Shakespeare example.

d) Follow similar steps for the data preparation as in the Shakespeare example. First extract the set of chars and define the encoding and decoding dictionaries. Then chop the data into X and y, you may define here a sequence length of 10 and a skip of 3. Finally transform your data and labels into one-hot vectors.

```
Hashplay Inc.
New Incentives
GrabJobs
MediBookr
MelissaWithLove.co
Starting 11
The CarShare Guy
Allahabad Bank
Anlaiye
Any Time Loan
Asieris Pharmaceuticals
Birner Dental Management Services (PERFECT TEETH)
Blockchain Foundry
Breethe
Buffalo Wild Wings
Canadian Solar
Convert Group
DeepCam
Doctaly
Gaze Coin
```

FIGURE 1 – 20 first entries of the training set of startup names

e) Define your model using a simple RNN layer (64 units) and a Dense layer with softmax activation layers.

f) Train your model.

g) Generate startup names using different values of the hyperparameters : number of epochs, number of cells in the RNN, etc.

Report about your findings in the pdf report. Select 5-10 generated startup names that you find interesting.

OPTIONAL - Many-to-many approach

Redo the previous exercise using this time a many-to-many approach. You will need to modify slightly the way the target tensors are prepared in order to do so. Pay also attention to the parameters of the layer definition that are a bit different.

OPTIONAL - Use GRU or LSTM

Replace the simple RNN layers with LSTM or GRU layers and comment on your observations :

— Can we reduce further the loss ?

— Do you observe improvements in the quality of generated names ?

Exercise 2 Word polarity detection with word embedding

The objective is to build a system that takes as input a word and outputs a probability that the word has a positive or negative connotation (*polarity*). As illustrated in Figure 2, the system is composed of two parts in which the word embedding part will leverage on pre-trained vectors such as the one obtained with word2vec or GloVe. If you are not familiar with word embeddings, we recommend you to watch the class material including videos and pdf posted on Moodle.

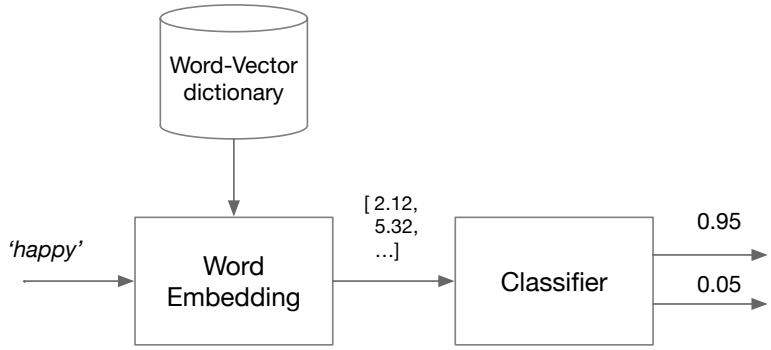

FIGURE 2 – Word polarity detection system composed of two parts : (1) the word embedding part using pre-trained dictionary mapping word to vector and (2) the classifier part.

a) **Read lexicons**. Download lists of positive and negative words from the *Opinion Lexicon* available from https://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html. A zip is also provided on Moodle. Listing 1 provides an example of function to read the lexicons. You may want to complete the code to remove lines that start with ;, that end with + and to remove empty lines. You should get 2005 positive words and 4783 negative words.

```
def read_vocabulary_from_file(filename):
    with open(filename, 'r', encoding="ISO-8859-1") as f:
        content = f.readlines() # content is a list of lines
        content = [x.strip() for x in content]  # removing newline chars
        ...
        return content
```

Listing 1 – Reading lexicon

b) **Convert words into vectors**. You can here go for two options : either by querying an online API that returns you the vectors for a given word, or download a pre-trained word-vector dictionary (word2vec, GloVe, etc.). The code provided in Listing 2 shows you how to realise this using a GloVe embedding – zip also provided on Moodle.

```
# to get GloVe vectors: wget http://nlp.stanford.edu/data/glove.6B.zip
def load_glove_embeddings(path):
    embeddings = {}
    with open(path, 'r', encoding='utf-8') as f:
```

```
        for line in f:
            values = line.strip().split()
            w = values[0]
            vectors = np.asarray(values[1:], dtype='float32')
            embeddings[w] = vectors
    return embeddings

# online query
import requests
import json
word = 'happy'
response = requests.get('https://icoservices.kube.isc.heia-fr.ch/word-
    embedding/wordvector/word2vec/en/' + word)
vector = response.json()

# off-line dictionary
word_dict = load_glove_embeddings('glove.6B/glove.6B.50d.txt')
word = 'happy'
vector = word_dict[word]   # if word is in word_dict
```

Listing 2 – Converting word to vectors

c) **Prepare the training and testing sets**. Prepare the tensors `X_train` for training by taking the corresponding vectors of 1500 positive and 1500 negative words from the lexicon. Prepare the `Y_train` target output tensor corresponding to the training set. You can, for example use the target [1.0, 0.0] for a positive word and [0.0, 1.0] for a negative word. For an embedding dimension of 50, the shapes of your `X_train` and `Y_train` tensors should be (3000,50) and (3000,2). In a similar way, prepare `X_test` and `Y_test` tensors.

d) **Train and evaluate a classifier**. Build a model, e.g. a double Dense layers in Keras (MLP) and train it. Report on the evolution of the loss and accuracy along the epochs. You should reach about 90% accuracy on the training set and 85% accuracy on the test set. Report on your model structure and fitting strategy.

e) **Analysis of results and discussions**. Report on your experiments and comment your classification performances. Find 5 to 10 words which are not in your training set and on which the system thinks it is clearly positive or negative. Find some words on which the system hesitates, i.e. with output probabilities in range [0.4-0.6]). Are these words and outputs making sense to you ?

f) **Optional – Extension to sentences**. Could you use the above system to make a polarity detection for the comments given by clients on products, e.g. on www.digitech.ch ? What would be the limitations of such system and how could we improve it ? What is happening when a word is not in the dictionary of the embedding ?

Exercise 3 Simple Seq2Seq Machine Translation

Optional : Warm-up

Read and try to re-run a basic Seq2Seq model in Keras to translate short English sentences into short French sentences, character-by-character. Note that it is fairly unusual to do character-level machine translation, as word-level models are more common in this domain.

Example using Keras functional API from :
https://blog.keras.io/a-ten-minute-introduction-to-sequence-to-sequence-learning-in-keras.html

Seq2Seq stacked LSTM in Pytorch

This exercise consists in completing the missing parts of the `seq2seq_pytorch` notebook file (available on moodle) that builds and trains a classical LSTM-based seq2seq problem for German to English translation, using PyTorch and torchtext.

Report and comment on your experiments.

Exercise 4 Optional : Review Questions

a) How can you define a generative system ? Describe the two approaches seen in the class to build generative systems with RNNs.

b) How can you define a generative system for sequence data ? Describe the two approaches seen in the class to build generative systems for sequence data.

c) Give two desired properties of word embeddings when used in a classification task (such as sentiment classification).

d) What are the three strategies to use an embedding layer in a deep system ?

e) Why is the system of exercise 1 working ?

f) What is the difference between Word2Vec and FastText ?

g) Draw the architecture of a Seq2Seq model and explain the encoder/decoder concept.

h) What is the main problem of a basic Seq2Seq model used for machine translation and the solution that can be used to overcome this problem ?