Lab 8: Shuffle Detection (Group)

You can do this lab in a **group** of 1-3 students of your choice. Note that the group size includes 1, so you can do this lab alone. However, working in a group is still highly recommended in order to brain-storm your approaches and possible glitches.
Working on this lab will be much easier after your individual RNN lab has been completed.

Due Day
Each our lab is included in a particular assignment as listed under "Assignment Schedule" on myPlace. Get in touch with the lecturer ASAP if you are unable to meet the deadline due to your individual circumstances or technical glitches on our lab computers.

How to Submit
The report for this lab needs to be uploaded to myPlace as a single PDF file using the submission link corresponding to the assignment in which this lab is included. Only one team member should upload the group report, but include the names of all the members. Use PDF format. Name the file "lab8.pdf"

Weight
This lab is worth 12% of the coursework.

Assessment Criteria
This lab will be assessed based on completion and correctness: if you obtain all the correct solutions, you will receive 100% mark. Partially correct or completed solutions will be discounted accordingly.

Tasks

(a) [50%]
Using our slides as guidance and the template provided (shuffle-detection-template.py), implement the fully connected model (ANN) that detects if a text sentence has been shuffled. Use the same corpus for training and testing as in our slides and in the template, but your model should use the first *3 words* (not just *2* as in our slides and template).
Hint: you can safely ignore the warning about softmax not automatically padding.
Make sure the accuracy on the test set is above 85%. Take the screenshots of your code and your output.

(b) [50%]
Using the definition of a simple Recurrent Neural Network (RNN) in our slides, and the Theano code you developed in your individual RNN lab, convert your program from (a) to using a simple RNN and apply to the same data. Make sure your model uses theano.scan, but not the ANN model from (a). For this, It needs to obtain the context vector for the word sequence (as in your individual RNN lab), and then use ANN (as in part(a)), to map that context vector to your prediction.
Make sure the accuracy on the test is above 90%. Take the screenshots of your code and your output.
Hint: use *theano.tensor.tanh* for non-linearity transformation in your recurrent step
Hint: for better results, you may need to initialize <u>all</u> your trainable parameters using normal distribution, e.g. by the following:

```
rng = numpy.random.RandomState(0)
```

```
vals = numpy.asarray(rng.normal(loc=0.0, scale=0.1, size=(n_words, word_embedding_size)))
word_embeddings  = theano.shared(vals, 'word_embeddings')
```

Hint: we may discuss additional hints and tips during lectures 15 and 16.