Project #2: NLP – Machine Translation – with Deep Learning Libraries

This project is composed of a number of components, which are sequential in the sense that you can go to one only after completing the priors. Each component has certain marks allocated as a percentage of the total.

Stage 1: Basic ANN and familiarization with Deep Learning Libraries

This assignment is an extension of project #1 where you have built up a fully connected Artificial Neural Network using the Python programming language utilizing its Numpy library and vectorization capabilities. You are to repeat the core activities of that assignment as described below, but this time generating the same program functionality using the Tensorflow high level library and possibly using Keras as a front-end software. It is your choice whether you would like to access Tensorflow through Keras, or directly.

You should learn Tensorflow on your own through material on the net, taking help from your classmates if needed. This will not be discussed in class. One possible good link is https://www.youtube.com/watch?v=PicxU81owCs&t=1s, which is from a Stanford Lecture Series made available on the public domain via YouTube. You can also download the PDF version of the same lecture. There are many other learning resources.

In this project, as in Project #1, you will build software for an Artificial Neural Network. You will train, validate and test your ANN on regression data of a *Combined Cycle Power Plant Dataset* given in http://archive.ics.uci.edu/ml/datasets/Combined+Cycle+Power+Plant which belongs to the UCI Machine Learning Data Repository. The given data consists of more than 9 thousand rows of 4 input variables and 1 output variable.

In project #1 you were asked to evaluate under what conditions of

1. ANN architecture
2. Training Data Granulation
3. Activation Functions and I/O Normalization
4. Weight Initialization
5. Learning Rate parameter
6. L2 Regularization parameter

you could achieve the best results. Using Tensorflow (and possibly Keras) you can run your code using the already-identified best combination of parameters.

For data splitting, use the same proportions as before, namely, 70 : 20 : 10 for training : validation : testing. While your training and validation data should be interspersed all across your given data set, your test data should be one or two continuous chunks from some end of your data set. Your program should be written such that every epoch of training is followed by a run over all the validation data samples so that you get both the training error and validation error at every epoch.

You should compare the results you obtained under these best-parameter combinations between the previous first-principles program and the current Tensorflow-based program, particularly the convergence histories and the test data errors.

You are free to experiment with more alternative-parameter values (other than your identified best combination) using the high-level libraries and report the same, drawing specific conclusions in the process.

You must also be aware that another DL high level library – Pytorch – is becoming more popular and likely exceeding Tensorflow. However, the next part of this project is structured to use Tensorflow only, so better to familiarise now with that.

Try to attain all computations both at this and at succeeding stages on your laptop CPUs. If that is difficult, try with Google Cloud, and then our inhouse Computation facilities. One small note explaining how to use Google Cloud is included in this package.

Your submission should be a sub-folder named Stage 1 containing your code, and a word doc (or PDF) containing the comparisons and conclusions mentioned above. **Credits: 30% of total project.**

NLP Stages:

These stages deal with the construction of a *Neural Machine Translator* using the Keras and Tensorflow frameworks that you are expected to have familiarized yourself with in the previous stage. Since these are computation-time intensive, the assignment starts with an easy and simple process which should be achievable by all. Your submissions should be sub-folders named Stage 2 and Stage 3, and possibly a Bonus Stage. The contents of these sub-folders are described in the following stages.

Stage 2: The included PDF is the final project assignment chapter of a book *Deep Learning for NLP* by Jason Brown Lee (you may search on the net); this is official and with permissions. It lays out step-by-step the process for translating simple sentences from a German Corpus (database location provided) into English. You are to meticulously follow the steps and you are free to use the Python code provided; and equally free to replace this with your own code – which action will, however, not earn any extra credits at this stage. Your submission at this stage will be the equivalent of Listing 30.28 and 30.29 (from Chap 30), using your own training and test outputs respectively, and importantly, two such output listings from training and test each. If your provided Python code (to project examiners) generates the same outputs, then you get full credits for this stage.

Credits: 40% of total project.

Stage 3: Stage 2 is an extremely simplified version, mainly to give you confidence in your ability to generate Machine Translations. One of the major simplifications (and source of inaccuracy) is that each output word in the target sentence is not sent as input to the following words, i.e. each target word depends only on the original source-language sentence. You are to modify the Python code (with Keras and TF) so that each target sentence word is now conditional not only on the source input sentence, but also on the preceding words of the target sentence. Note that needs modification of both training and testing (i.e. forward calculating) parts of the code. As deliverables, you are again to generate the 4 listings (incl. Bleu scores), marking them as outputs from Stage 3.

Credits: 30% of total project.

Bonus Stage : Stage 2 was the Greedy search version. Now you are to extend this to BEAM search. Take a BEAM width of 3. [A Beam width of *k* will increase computation time by a factor of *k*, if you are ready to take more, do so]. Remember BEAM search is at the decoder stage for the *forward calculations alone*. If you want to look into BEAM search code, you can see (and download from) https://gist.github.com/nikitakit/6ab61a73b86c50ad88d409bac3c3d09f . You need to login to Github

– you are advised to create an account. I have these two downloaded Python codes, but prefer that you download directly. You are also allowed to use patches from code in Github. As deliverables, you are again to generate the 4 listings (incl. Bleu scores), marking them as outputs from Bonus Stage. Further, you must provide the code for Bonus Stage, and your provided listings will be compared with the outputs from the code you have provided.

Credits: 10% extra.