Deep Learning Main Project 2020

Value: 60% - Due Date Friday 1st May

# Overview

This is a revised project specification for the 2020 delivery of the Deep Learning class. The revised project specification has been developed in light of the Covid-19 situation. This project specification differs from the original outline I gave verbally in our last physical class on the 10th of March. For example, there was an original plan to have a two stage submission; this has been dropped to simplify the process.

The goal of the project is to perform a systematic investigation of a number of Deep Learning methods in the context of text processing tasks and benchmark these methods against classical methods where appropriate.

This project has been designed to give you a range of task elements that you can use to build up your skills in Deep Learning. As such this project specification is detailed and descriptive.

The outputs for this project will be a detailed report (minimum 11 pages formatted as per this document), source code, and a trained model. Detail on what is to be included in each of these pieces is detailed below. The report is to be submitted via Brightspace with links to a code archive (.zip or .tar.gz), and a link to two trained models (described below).

Your implementation should be made in Keras / TensorFlow and you cannot use any alternative data set.

# Task Specification

## Part 1: IMDB Modelling Task - 50% (of project total)

The core of this project is based around a simple task -- performing sentiment analysis with the IMDB dataset given here:

<https://www.kaggle.com/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews>

There are 50,000 documents in the IMDB corpus. Split these into the following ratio for analysis:

* Train: 50%
* Validation: 30%
* Test: 20%

We have already done a lot with this dataset, so this assignment should at its core naturally allow you to move beyond our notes. Below I’ve set out a number of different comparisons that you are to perform. You can think of these as being individual chunks of analysis. These do not necessarily build on each other. It is up to you to decide whether these are done in isolation or not. The key point is that you understand yourself what you are doing, and that this understanding is reflected both in well documented code, and in your report.

Your models should be designed to minimize overfitting as appropriate.

In all cases you should record your results as graphs for Training and Validation data and report the test result after training has been completed. Please select whatever less functions or metrics you think are appropriate based on notes provided in class, or more widely based on what is appropriate in a text classification task.

**RNN Variants**

The first sub-assignment is to compare performance on the classification task across Recurrent Network Variants. Specifically compare LSTM and Basic RNN models. You are free to choose your own state size for the recurrent network, however please use the same state size for both RNN variants.

Also compare a single layer LSTM implementation to a multi-layer LSTM implementation.

**Embeddings**

Distributed embeddings provide a lot of power in text classification, but there are many different Embeddings types that can be used. Compare classification between Embeddings learned on the fly to any pre-trained word embedding available from the Tensorflow Hub.

**CNN for Text Classification**

As mentioned in the lecture notes, CNNs are designed to model local features while LSTMs are very good at handling long range dependencies. Investigate the use of CNNs with multiple and heterogeneous kernel sizes both as an alternative to an LSTM solution, and as an additional layer before a LSTM solution.

**Model Saving**

From the various models above, save the best model. A link to the best performing model should be included in your submission. You will also be using this saved model in Part 3 below. There are many ways in which models can be saved. I’m not prescribing a specific way this is to be done. You are free to use whichever method you find most suitable. As always clearly document your design.

## Part 2: Working with your own Data - 25%

One problem with libraries which provide wrappers for well-known datasets is that they can make the task of using the dataset so easy, that we do not realise what is required in the construction and use of data in Deep Learning. Related to this, in real world problems you will have your own data and will often want to build on pre-trained models to make use of the learning that has already been achieved with an existing model -- doing this is called Transfer Learning.

Given these issues, in this part of the assignment you will collect your own dataset and use it to train a model that is based on your own existing pre-trained model constructed in Part 1.

**Data Collection**

Your first task is to construct a labelled dataset and encode so that it can be used again for processing. We will adopt the actual IMDB movie database as the source of information -- not the dataset used in Part 1.

To do this, randomly select 30 movies from the year of your birth. You can use IMDB’s title based search functionality to do this.

<https://www.imdb.com/search/title/>

For each movie, select at least one good and one bad review. The reviews should be in English, and I’ll let you decide what is a good and a bad review, but for instance a good review might be 7/10 stars or higher and a bad review might be 4/10 stars or lower.

For each review that you select, record whether this was a positive or negative review and also capture the text for that review. You will probably get better results if you include the title of the review in the document you record for that review. How you record your reviews initially is up to you, but options include an excel file, a CSV file, a set of individual text files. It is very much up to you. Keep in mind though that the raw data will have to be supplied along with your source code.

**Modelling**

Using the best performing model from Part 1, load the data -- split 70/30 between training and validation -- no testing data as you don’t have enough. Then, build a model for this new novel data that is based on a previously trained model from Part 1. This implies that you need to fine tune the existing model to your new data and test its performance. This does not mean that you add your data to the original IMDB data. This fine tuned model should start from the model that you saved in Part 1.

Report Training and Validation scores for this fine-turned model. Save this model. It needs to be supplied along with the original model it is based on as part of your assignment.

Finally, build a “from scratch” model for your novel data that uses the exact same architecture as your best performing model from Part 1. Compare the performance of this “from scratch” model to your fine-tuned pre-trained model.

## Part 2: Writing your own Reviews - 25%

Practical language processing tasks aren’t just about classifying. In the second part of the assignment you will put your skills in RNNs and related technologies to work to generate some original text and benchmark your model against a more classical implementation.

For this work make use of the IMDB dataset but let’s split the data differently. For each of the examples below, build one model with negative reviews, one model with positive reviews, and one model with all reviews included. Keep in mind that we do not need to use a training validation and testing split of the data in this case.

Your core model should be based on the use of LSTMs, but beyond this you are free to explore whatever architecture and hyper-parameter variants that you find results in the best performance in the language generation task.

With the same data, also implement a statistical model for language generation.

Report model performance in terms of perplexity. Provide 5 outputs each from your best implementation and the statistical model. Make sure to save your best model and provide it vai a link in the submission.

# Document Specification

Your document should be at least 11 pages in length and formatted as per the current document, i.e., standard margins, 11pt Arial font, 1.15 spacing between lines, and a 50% spacing (approximately) between paragraphs.

The first page is a standalone cover page. The cover page should include:

* Your name
* Your student number
* A link to a zip file that contains all your code (not models) collected data files etc.
* A link to a file that contains the three trained models listed in Part 1 and Part 2.
* The following statement:

*“I confirm that the document and related code here is all my own work and that I did not engage in unfair practice in any way.”*

This report should focus on the results and your analysis -- not on a background explanation of Deep Neural Networks or related models.

The following sections should be included:

* IMDB Modelling Task
* Working with my own Data
* Writing my own Reviews
* Discussion

Appendices can be included, but these will not be counted in the 11 page minimum submission.

# Submission Details

This assignment is due for submission at 10pm on Friday 1st May. Extensions will not be granted except in the case of documented and approved personal circumstances. Late submissions will be deducted 2 percent per day for each of the first two days late, and then 1 percent per day for each day thereafter until an absolute deadline of May 20th.

All students should be available to answer questions on their assignment on Thursday 14th and Friday 15th of May. Arrangements for this will be made later, but will likely be by appointment on Zoom -- accessible by either phone or online.

# Marking Rubric

Marks for Parts 1, 2 and 3 of this assignment are split in the ratio 50%, 25%, 25%. For each Part, marks will in turn be broken down in a split between documentation and model development. The Rubric below details how marks will be split out of 100 for each of the three parts.

0 - 19: The student failed to provide a working implementation of code for achieving this part of the assignment -- or an unacceptable amount of code or documentation indicated unfair practice.

20 - 39: The student provided a working implementation of this Part of the assignment, but the documentation or interview failed to show the student had a clear understanding of the methods employed.

40 - 59: The student provided a working implementation of this Part of the assignment, and the documentation or interview showed that the student had a clear understanding of the methods employed.

60 - 79: The student provided a high quality implementation of this part of the assignment with a high quality report that was detailed in analysis and well motivated.

80 - 100: The student provided an excellent implementation of this Part of the assignment with a report of quality that is worthy for submission for publication at a national conference. To reach this grade the work will generally need a non-trivial original element that is not already covered by the assignment specification -- this is in addition to meeting the other criteria.

All students should be able to defend / explain their submission orally if requested to do so. Any failure to do so will be judged as a failure in the documentation to show the student had a clear understanding of the methods employed.

Unfair practice of any type is not accepted. Do not copy text into your report without appropriate citations. Even with citations, no more than 10% of a report should be based around existing material. Note that this also includes ‘self plagiarism’. Similarly, no copying blocks of code into your code without appropriate acknowledgement. Again, even with acknowledgements, only a small portion of code should be copied directly. Unfair practice of this type will be subject to a grade of 0 on the assignment. If you are in doubt - ask me.