FIT3181 Deep Learning – S2 2023

Assignment 2 – Deep Learning for Sequential Data

Purpose	This assignment focuses on machine learning concepts and practical skills in building and training deep neural networks for sequential data, including Recurrent Neural Networks (RNNs) and Transformers. You are expected to demonstrate a comprehensive understanding of RNNs and Transformers, and ultimately put them into practice by constructing models for a text classification task. The assignment relates to Unit Learning Outcomes 1, 2, 3, 4, 5, and 6.
Your task	Complete the individual tasks as detailed in the instructions below.
Value	**20%** of your total marks for the unit The assignment is marked out of 100 marks.
Due Date	**11:55 pm Friday 20 October 2023**
Submission	● Via Moodle Assignment Submission. ● Turnitin will be used for similarity checking of all submissions. ● This is an individual assignment (group work is not permitted). ● DRAFT submission is not assessed. ● **In this assessment, you must not use generative artificial intelligence (AI) to generate any materials or content in relation to the assessment task.**
Assessment Criteria	Marks are awarded for the *understanding* and *correctness* of your work, including but not limited to your code, calculations, and explanations to respond to required tasks.
Late Penalties	**20%** deduction per calendar day
Support Resources	See Moodle Assessment page
Feedback	Feedback will be provided on student work via: ● general cohort performance ● specific student feedback ten working days post-submission

INSTRUCTIONS

This assignment has two (2) sections:

- Section 1 consists of knowledge questions to assess your understanding of the advantages of CNN, RNNs, and Transformers, and how the forward propagation and backward propagation in RNNs work.
- Section 2 consists of coding questions on Word2Vect, Text CNN, RNNs, and Transformers. In Section 2, you are required to build, train, test, and improve deep learning models for the text classification task. You are also required to provide your comments on the results of your models.

Make sure you read the instructions carefully.

You need to **submit through the Moodle** Assignment activity a single ZIP file, name **xxx_A2_solution.zip**, where xxx is your student ID. The ZIP file should contain:

1) A pdf file with answers to the questions in Section 1. It should be named **A2_Section1_Solutions.pdf**

2) A Jupyter notebook with answers to the questions in Section 2. It should be named **A2_Section2_Solutions.ipynb**.

3) A copy of your solution notebook for Section 2 exported in HTML format.

4) Your best model (see Question 4.3). In case your best model is too heavy to be submitted to the unit Moodle site, you need to provide us with the link to download it.

5) Any extra files or folders needed to complete your assignment.

Section 1: Knowledge Questions (15 marks)

Question 1 (1 mark)

Under which learning scenarios should we combine both CNNs and RNNs (Hint: types of input data)? You need to provide explanations to justify your answer and provide at least two example problems that you would combine CNNs and RNNs to solve. Your answer must be less than 150 words in total.

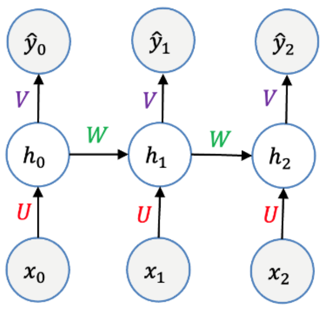
Question 2 (2 marks)

What are the main advantages of Transformers over RNNs? You need to provide explanations to justify your answer. Your answer must be less than 200 words in total. For further information on the Transformer model, refer to **this paper**.

Question 3 (12 marks)

This question is about the computation in a basic RNN.

Assume that we feed an input sequence x_0, x_1, x_2 with the ground-truth labels y_0, y_1, y_2 to a basic RNN with 3 hidden states for classification task as shown in Figure 1. The activation function used when calculating hidden states is the `tanh` function. The values of the input sequence, ground-truth labels, and network parameters are given below. The notations are as described in lecture 6 (see slide 25).



$$x_0 = \begin{bmatrix} 1 \\ -1 \\ 0 \end{bmatrix} \quad y_0 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \quad U = \begin{bmatrix} 1 & 2 & 3 \\ -1 & 0 & 1 \\ 2 & -1 & 0 \end{bmatrix} \quad b = \begin{bmatrix} 0 \\ 1 \\ -1 \end{bmatrix}$$

$$x_1 = \begin{bmatrix} -1 \\ 1 \\ 0 \end{bmatrix} \quad y_1 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \quad W = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 1 & 0 \\ -1 & 2 & 1 \end{bmatrix} \quad c = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$$

$$x_2 = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} \quad y_2 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \quad V = \begin{bmatrix} 2 & -1 & 0 \\ 1 & 1 & -1 \\ -1 & 2 & 1 \end{bmatrix}$$

Figure 1. A basic RNN for classification task.

$\bar{h}_0 = Ux_0 + b, h_0 = tanh(\bar{h}_0), \hat{y}_0 = softmax(Vh_0 + c)$.

For $t > 0$, $\bar{h}_t = Wh_{t-1} + Ux_t + b$, $h_t = tanh(\bar{h}_t), \hat{y}_t = softmax(Vh_t + c)$.

Question 3.1 (1.5 marks)

Compute the numerical values of h_0, h_1, h_2.

Question 3.2 (1.5 marks)
Compute the numerical values of $\hat{y}_0, \hat{y}_1, \hat{y}_2$.

Question 3.3 (2 marks)
Given that the loss between the ground-truth label and the predicted label at each time step is the cross-entropy (CE) loss, compute the numerical values of losses
$l_0 = CE(y_0, \hat{y}_0), l_1 = CE(y_1, \hat{y}_1), l_2 = CE(y_2, \hat{y}_2)$ and the total loss $L = \frac{1}{3}(l_0 + l_1 + l_2)$.

Question 3.4 (2 marks)
Compute the gradient $\frac{\partial L}{\partial h_1}$.

Question 3.5 (2 marks)
Compute the gradient $\frac{\partial L}{\partial V}$.

Question 3.6 (3 marks)
Compute the gradient $\frac{\partial L}{\partial U}$.

Note: For questions 3.4, 3.5, and 3.6, you must show both formulas and numerical results to get full marks.

Section 2: Coding Questions (85 marks)

The task we are considering in this section is the text classification task. A notebook has been provided (**A2_Section2_Student_notebook.ipynb**), and you are required to provide your answers in the notebook.

The dataset we use for this assignment is a question classification dataset for which the training set consists of $5{,}500$ questions belonging to 6 coarse question categories including:

- abbreviation (ABBR),
- entity (ENTY),
- description (DESC),
- human (HUM),
- location (LOC) and
- numeric (NUM).

Here is the [link] to the dataset. In this assignment, we will utilize a subset of this dataset, containing 2,000 questions for training and validation. We will use 80% of those 2000 questions for training and the rest for validation.

Section 2 consists of 5 parts:

- Part 0: Downloading and preprocessing data. This part has been completed. No marks are allocated for this part.
- Part 1: Coding assessment on using Word2Vect to transform texts to vectors (20 marks).
- Part 2: Coding assessment on Text CNN for sequence modeling and neural embedding (10 marks).
- Part 3: Coding assessment on RNNs for sequence modeling and neural embedding (32 marks).
- Part 4: Coding assessment on Transformer for sequence modeling and neural embedding and the overall ranking (23 marks).

For each cell marked with **# Insert your code here** in the notebook, these are placeholders where you must supply your own codes when instructed.

Part 0: Download and preprocess the data

This part has been completed. No marks are allocated for this part.

Preprocessing data is a crucial initial step in any machine learning or deep learning project. The *TextDataManager* class simplifies the process by providing functionalities to download and preprocess data specifically designed for the subsequent questions in this assignment. It is highly recommended to gain a comprehensive understanding of the class's functionality by **carefully reading** the content provided in the *TextDataManager.py* file before proceeding to answer the questions in Part 1.

The following parts summarize the questions. The detailed questions are in the provided notebook.

Part 1: Using Word2Vect to transform texts to vectors (20 marks)

In this part, you are required to use a pretrained Word2Vect model for realizing a machine learning task. Basically, you will use a pretrained Word2Vect model to transform the questions in the above dataset to numeric form for training a classifier using Logistic Regression. You are also required to visualize embedding vectors.

Question 1.1 (2 marks)

Write code to download the pretrained model ***glove-wiki-gigaword-100.*** Write code for the function ***get_word_vector(word, model)*** used to transform a word to a vector using the pretrained Word2Vect model ***model***.

Question 1.2 (2 marks)

Write code for the function `get_sentence_vector(sentence, important_score=None, model=None)`. This function will transform a sentence to a 100-dimensional vector using the pretrained model ***model***.

Question 1.3 (4 marks)

Write code to transform questions to feature vectors.

Question 1.4 (2 marks)

Use ***MinMaxScaler(feature_range=(-1,1))*** in scikit-learn to scale both training and validation sets to the range **(−1,1)**.

Question 1.5 (2 marks)

Train a Logistic Regression model on the training set and then evaluate it on the validation set.

Question 1.6 (3 marks)

Write code for the **`function find_most_similar(word=None, k=5, model=None)`** which returns a list of the top-**`k`** most similar words (in descending order) for a given word. Similarity is measured by the cosine similarity of embedding vectors obtained from the pretrained Word2Vect model **`glove-wiki-gigaword-100`**.

Question 1.7 (5 marks)

Implement the **`plot2D_with_groups(word_list, model, k=10)`** function to visualize groups of similar words in 2D space. The **`word_list`** parameter is a list of words, and for each word in the **`word_list`**, find its top-**`k`** most similar words (which forms a group) using the **`find_most_similar`** function. Use tSNE to project embedding vectors into 2D space and plot groups with different colors.

Part 2: Text CNN for sequence modeling and neural embedding (10 marks)

In this part, you are required to implement the Text CNN model, followed by the training and testing of the model using the mentioned dataset. The paper of Text CNN can be found at this [link].

Question 2.1 (8 marks)

Complete the code for the TextCNN class.

Question 2.2 (2 marks)

Complete the code to test the Text CNN model.

Part 3: RNN-based models for sequence modeling and neural embedding (32 marks)

In this part, you are required to construct RNNs which are trained and tested using the mentioned dataset. Specifically, you are required to construct (3.1) RNNs with different cell types, i.e., *simple_rnn*, *gru*, and *lstm*, (3.2) RNNs with fine-tuning embedding matrix, and (3.3) RNNs with attention.

3.1. RNNs with different cell types (12 marks)

Question 3.1.1 (7 marks)

Complete code to construct a vanilla RNN. The cell type can take one of the three values: ‘*simple_rnn*’, ‘*gru*’ or *‘lstm’.*

Question 3.1.2 (1 mark)

Complete code to run the constructed RNN with simple RNN ('simple_rnn') cell.

Question 3.1.3 (1 mark)

Complete code to run the constructed RNN with GRU ('gru') cell.

Question 3.1.4 (1 mark)

Complete code to run the constructed RNN with LSTM ('lstm') cell.

Question 3.1.5 (2 marks)

Write code to conduct experiments to compare the accuracies of RNNs with the three different cell types and give your comments on the results.

3.2. RNNs with fine-tuning embedding matrix (8 marks)

Question 3.2.1 (6 marks)

Extend the class BaseRNN in Part 3.1 to achieve a class RNN in which the embedding matrix can be initialized using a pretrained Word2Vect.

Question 3.2.2 (2 marks)

Write code to conduct experiments to compare three running modes for the embedding matrix and give your comments on the results.

3.3. RNNs with Attention for Text and Sequence Classification (12 marks)

Question 3.3.1 (4 marks)

Complete code for the class `MyAttention` inherited from `tf.keras.layers.Layer`.

Question 3.3.2 (6 marks)

Extend the class `RNN` in Question `3.2.1` to achieve the class `AttentionRNN`.

Question 3.3.3 (2 marks)

Choose a common setting for standard RNN and RNN with attention and conduct experiments to compare them. Give your comments on the results.

Part 4: Transformer-based models for sequence modeling and neural embedding and the overall ranking (23 marks)

In this part, you are required to implement the multi-head attention module of the Transformer for the text classification problem. You are also required to conduct experiments to evaluate the impact of hyperparameters.

For the overall ranking, for any models defined in the previous questions of all parts, you are free to fine-tune hyperparameters such that you get the best model. The assessment will be based on your best model's accuracy, with up to 9 marks available.

Question 4.1 (11 marks)

Implement the multi-head attention module of the Transformer for the text classification task. In this part, we only use the output of the Transformer encoder for the task.

Question 4.2 (3 marks)

Write code to conduct experiments to evaluate the impact of hyperparameters of the Transformer model on the accuracy. Report your findings from the experiments. The necessary condition to get the full mark for this question is that the accuracy of your best Transformer model should be at least 90%.

Question 4.3 (9 marks)

For any models defined in the previous questions of all parts, you are free to fine-tune hyperparameters such that you get the best model, i.e., the model with the highest accuracy on the validation set. The assessment will be based on your best model's accuracy, with up to 9 marks available.

END OF ASSIGNMENT

GOOD LUCK WITH YOUR ASSIGNMENT 2!