Assignments

**M3224.000300 Natural Language Processing with Neural Networks**

Fall 2023

Graduate School of Data Science
Seoul National University
**Instructor: Dr.Jay-yoon Lee (lee.jayyoon@snu.ac.kr)**
**TA: Hye Ryung Son (hyeryung.son@snu.ac.kr)**
**TA: Seongil Park (athjk3@snu.ac.kr)**

**Due on Tuesday Oct. 24, 2023 by 9:30am (before class)**

# (Assignment 3: Recurrent Neural Networks)

**Honor Pledge for Graded Assignments**

"I, YOUR NAME HERE , affirm that I have not given or received any unauthorized help on this assignment, and that this work is my own."

## 0 Instructions

- Total score cannot exceed 100 points. For example, if you score 98 points from non-bonus questions and 3 points are added from bonus questions, your score will be 100 points, not 101 points.
- Skeleton codes for problem 2 and 3 are at the directory `/q2` and `/q3` each. Problem 1 does not have coding problems.
- Run the `bash collect_submission.sh` script to produce your 2000_00000_coding.zip file. Please make sure to modify collect_sumbssion.sh file before running this command. (**2000_00000** stands for your student id)
- Modify this tex file into `2000_00000_written.pdf` with your written solutions
- Upload both `2000_00000_coding.zip` and `2000_00000_written.pdf` to etl website.
- **If submission instructions are not followed, `4 points` will be deducted.**

## 1 NLP tasks with RNN (15 pts)

RNNs are versatile! In class, we learned that this family of neural networks have many important advantages and can be used in a variety of tasks. They are commonly used in many state-of-the-art architectures for NLP.

For each of the following tasks, state how you would run RNN to do that task. In particular, specify how the RNN would be used at test time (not training time), and specify

- How many outputs i.e. number of times the softmax $\hat{y}^{(t)}$ is called from your RNN. If the number of outputs is not fixed, state it as arbitrary.
- What each $\hat{y}^{(t)}$ is a probability distribution over (e.g. distributed over all species of categories)
- Which inputs are fed to produce each output $\hat{y}^{(t)}$

The inputs for each of the tasks are specified below.

(a) **Movie Rating (3pts)**: Classify sentiment of a movie review ranging from negative to positive (integer values from 1 to 5).
Inputs: A sentence containing n words. **Answer:**

(b) **Part-of-speech Tagging (4pts)**: For each word in a sentence, categorize that word in correspondence with a particular part-of-speech such as either nouns, verbs, adjectives, adverbs, etc.
Inputs: A sentence containing n words. **Answer:**

(c) **Text Generation (4pts)**: Generate text from a chatbot that was trained to speak like a news anchor by predicting the next word in the sequence.
Input: A single start word or token that is fed into the first time step of the RNN. **Answer:**

(d) **Machine Translation (4pts)**: Translate the given sentence into another language.
Input: A sentence containing n words. **Answer:**

## 2 Topic Classfication with RNN (35 pts)

In this assignment, we will implement RNN model for topic classification task using AG news dataset. Topic classification is a task that involves classifying text into different categories or subjects.
For this question, please update the given jupyter notebook file **(q2/topicClassification.ipynb)** and submit it along with your answer to this latex file. Please note that this assignment is built and tested under Google Colaboratory with **T4 GPU** (available for free). If you work on a local machine, you need to handle version issue on your own.

### 2.1 Implementing an RNN model with PyTorch (15 pts)

In this question, we are going to implement a SimpleRNN class to classify AG news dataset. Complete the code following the instruction in the jupyter notebook file.

(a) Implement the `__init__()` function of SimpleRNN class. **(5 pts)**

(b) Implement the `forward()` function of SimpleRNN class. **(10 pts)**

### 2.2 Train and evaluate the RNN model (20 pts)

Next, we will train and evaluate the RNN model. Implement a train and evaluation code following the instruction in jupyter notebook file.

(a) Implement the `train()` function of `Trainer` class. **(8 pts)**

(b) After the training is complete, use the `plot()` function of `Trainer` class to display the figure, and then paste it here. **(4 pts)**

(c) Report the best validation accuracy using the `print_best_acc()` function of `Trainer` class. **(2 pts)**

(d) Based on (b) and (c), evaluate whether the training was successful, and write at least two ways to improve the model's performance. **(6 pts)**

## 3 Neural Machine Translation with LSTM (50+3)

In Nerual Machine Translation (NMT), our goal is to convert a sentence from the *source* language to the *target* language. In this assignment, we will implement a sequence-to-sequence (Seq2Seq) network with attention, to build a Neural Machine Translation (NMT) system between Jeju dialect and Korean. In this section, we describe the **training procedure** for the proposed NMT system, which uses a Bidirectional LSTM Encoder and a Unidirectional LSTM Decoder. After training, you can find out how the NMT system is better than you in translating Jeju dialect. The model is trained and evaluated on JIT (Jejueo interview transcripts) dataset [1]

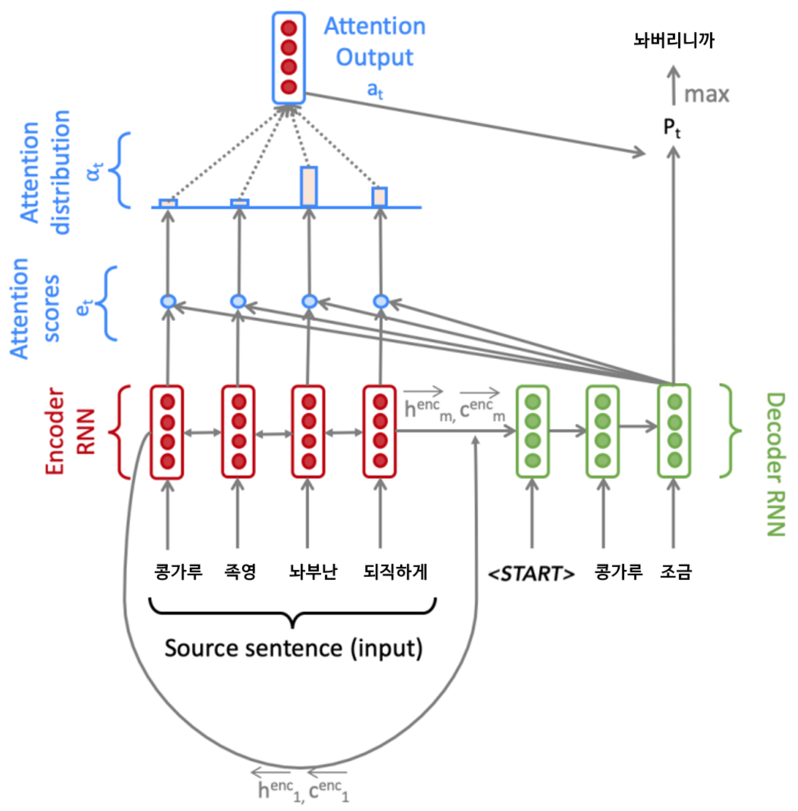

Figure 1: Seq2Seq Model with Multiplicative Attention, shown on the third step of the decoder. (NOTE: Embedding of NMT in the assignment differs from described above.)

## 3.1 Training Procedure

Given a sentence in the source language (Jeju dialect), we look up the subword embeddings from an embeddings matrix, yielding $\mathbf{x}_1, \dots, \mathbf{x}_m$ ($\mathbf{x}_i \in \mathbb{R}^{e \times 1}$), where $m$ is the length of the source sentence and $e$ is the embedding size. We feed these embeddings to the bidirectional encoder, yielding hidden states and cell states for both the forwards ($\rightarrow$) and backwards ($\leftarrow$) LSTMs. The forwards and backwards versions are concatenated to give hidden states $\mathbf{h}_i^{enc}$ and cell states $\mathbf{c}_i^{enc}$:

$$\mathbf{h}_i^{\text{enc}} = [\overleftarrow{\mathbf{h}_i^{enc}}; \overrightarrow{\mathbf{h}_i^{enc}}] \text{ where } \mathbf{h}_i^{\text{enc}} \in \mathbb{R}^{2h \times 1}, \overleftarrow{\mathbf{h}_i^{enc}}, \overrightarrow{\mathbf{h}_i^{enc}} \in \mathbb{R}^{h \times 1} \qquad 1 \le i \le m \qquad (1)$$

$$\mathbf{c}_i^{\text{enc}} = [\overleftarrow{\mathbf{c}_i^{enc}}; \overrightarrow{\mathbf{c}_i^{enc}}] \text{ where } \mathbf{c}_i^{\text{enc}} \in \mathbb{R}^{2h \times 1}, \overleftarrow{\mathbf{c}_i^{enc}}, \overrightarrow{\mathbf{c}_i^{enc}} \in \mathbb{R}^{h \times 1} \qquad 1 \le i \le m \qquad (2)$$

We then initialize the decoder's first hidden state $\mathbf{h}_0^{\text{dec}}$ and cell state $\mathbf{c}_0^{\text{dec}}$ with a linear projection of the encoder's final hidden state and final cell state.[2]

$$\mathbf{h}_0^{\text{dec}} = \mathbf{W}_h[\overleftarrow{\mathbf{h}_1^{enc}}; \overrightarrow{\mathbf{h}_m^{enc}}] \text{ where } \mathbf{h}_0^{\text{dec}} \in \mathbb{R}^{h \times 1}, \mathbf{W}_h \in \mathbb{R}^{h \times 2h} \qquad (3)$$

$$\mathbf{c}_0^{\text{dec}} = \mathbf{W}_c[\overleftarrow{\mathbf{c}_1^{enc}}; \overrightarrow{\mathbf{c}_m^{enc}}] \text{ where } \mathbf{c}_0^{\text{dec}} \in \mathbb{R}^{h \times 1}, \mathbf{W}_c \in \mathbb{R}^{h \times 2h} \qquad (4)$$

With the decoder initialized, we must now feed it a target sentence. On the $t^{th}$ step, we look up the embedding for the $t^{th}$ subword, $\mathbf{y}_t \in \mathbb{R}^{e \times 1}$. We then concatenate $\mathbf{y}_t$ with the *combined-output vector* $\mathbf{o}_{t-1} \in \mathbb{R}^{h \times 1}$ from the previous timestep (we will explain what this is later down this page!) to produce $\overline{\mathbf{y}_t} \in \mathbb{R}^{(e+h) \times 1}$. Note that for the first target subword (i.e. the start token) $\mathbf{o}_0$ is a zero-vector. We then feed $\overline{\mathbf{y}_t}$ as input to the decoder.

[1]https://www.kaggle.com/datasets/bryanpark/jit-dataset

[2]If it's not obvious, think about why we regard $[\overleftarrow{\mathbf{h}_1^{enc}}, \overrightarrow{\mathbf{h}_m^{enc}}]$ as the 'final hidden state' of the Encoder.

$$\mathbf{h}_t^{\text{dec}}, \mathbf{c}_t^{\text{dec}} = \text{Decoder}(\overline{\mathbf{y}_t}, \mathbf{h}_{t-1}^{\text{dec}}, \mathbf{c}_{t-1}^{\text{dec}}) \text{ where } \mathbf{h}_t^{\text{dec}} \in \mathbb{R}^{h\times 1}, \mathbf{c}_t^{\text{dec}} \in \mathbb{R}^{h\times 1} \tag{5}$$
$$\tag{6}$$

We then use $\mathbf{h}_t^{\text{dec}}$ to compute multiplicative attention over $\mathbf{h}_1^{\text{enc}}, \dots, \mathbf{h}_m^{\text{enc}}$:

$$\mathbf{e}_{t,i} = (\mathbf{h}_t^{\text{dec}})^T \mathbf{W}_{\text{attProj}} \mathbf{h}_i^{\text{enc}} \text{ where } \mathbf{e}_t \in \mathbb{R}^{m\times 1}, \mathbf{W}_{\text{attProj}} \in \mathbb{R}^{h\times 2h} \qquad 1 \le i \le m \tag{7}$$
$$\alpha_t = \text{softmax}(\mathbf{e}_t) \text{ where } \alpha_t \in \mathbb{R}^{m\times 1} \tag{8}$$
$$\mathbf{a}_t = \sum_{i=1}^{m} \alpha_{t,i} \mathbf{h}_i^{\text{enc}} \text{ where } \mathbf{a}_t \in \mathbb{R}^{2h\times 1} \tag{9}$$

$\mathbf{e}_{t,i}$ is a scalar, the $i$th element of $\mathbf{e}_t \in \mathbb{R}^{m\times 1}$, computed using the hidden state of the decoder at the $t$th step, $\mathbf{h}_t^{\text{dec}} \in \mathbb{R}^{h\times 1}$, the attention projection $\mathbf{W}_{\text{attProj}} \in \mathbb{R}^{h\times 2h}$, and the hidden state of the encoder at the $i$th step, $\mathbf{h}_i^{\text{enc}} \in \mathbb{R}^{2h\times 1}$.

We now concatenate the attention output $\mathbf{a}_t$ with the decoder hidden state $\mathbf{h}_t^{\text{dec}}$ and pass this through a linear layer, tanh, and dropout to attain the *combined-output* vector $\mathbf{o}_t$.

$$\mathbf{u}_t = [\mathbf{a}_t; \mathbf{h}_t^{\text{dec}}] \text{ where } \mathbf{u}_t \in \mathbb{R}^{3h\times 1} \tag{10}$$
$$\mathbf{v}_t = \mathbf{W}_u \mathbf{u}_t \text{ where } \mathbf{v}_t \in \mathbb{R}^{h\times 1}, \mathbf{W}_u \in \mathbb{R}^{h\times 3h} \tag{11}$$
$$\mathbf{o}_t = \text{dropout}(\tanh(\mathbf{v}_t)) \text{ where } \mathbf{o}_t \in \mathbb{R}^{h\times 1} \tag{12}$$

Then, we produce a probability distribution $\mathbf{P}_t$ over target subwords at the $t^{th}$ timestep:

$$\mathbf{P}_t = \text{softmax}(\mathbf{W}_{\text{vocab}} \mathbf{o}_t) \text{ where } \mathbf{P}_t \in \mathbb{R}^{V_t\times 1}, \mathbf{W}_{\text{vocab}} \in \mathbb{R}^{V_t\times h} \tag{13}$$

Here, $V_t$ is the size of the target vocabulary. Finally, to train the network we then compute the cross entropy loss between $\mathbf{P}_t$ and $\mathbf{g}_t$, where $\mathbf{g}_t$ is the one-hot vector of the target subword at timestep $t$:

$$J_t(\theta) = \text{CrossEntropy}(\mathbf{P}_t, \mathbf{g}_t) \tag{14}$$

Here, $\theta$ represents all the parameters of the model and $J_t(\theta)$ is the loss on step $t$ of the decoder. Now that we have described the model, let's try implementing it for Jeju dialect to Korean translation!

## 3.2 Setting up Virtual Environment

In this part, we will set up a virtual environment for implementing the NMT machine. Before you begin, make sure to read instructions about using GSDS cluster on the ETL board. Run the following commands within the assignment directory (`a3`) to create the appropriate conda environment. This guarantees that you have all the necessary packages to complete the assignment. You will be asked to implement LSTM cells and the seq2seq model using the PyTorch package.

```
conda create -n a3q3
conda activate a3q3
bash env.sh
conda activate a3q3
```

## 3.3 Implementation Questions

(a) To ensure the sentences in a given batch are of the same length, we must pad shorter sentences to be the same length after identifying the longest sentence in a batch. Implement the `pad_sents` function in `utils.py`, which returns padded sentences. **(3 pts)**

(b) Implement the code of class `LSTMCell` in the file `assignment_code.py`. LSTMCell contains two functions: initialization `__init__()` and forward `forward()`. You can refer to the PyTorch documentation[3] or GRUcell class which is implemented on the skeleton code. **(8 pts)**

(c) Implement the `__init__` function in `model_embeddings.py` and `nmt_model.py` to initialize the necessary model embeddings and layers for the NMT system. You can run sanity check by executing `python sanity_check.py 1c` **(3 pts)**

(d) Implement the `encode` function in `nmt_model.py`. This function converts the padded source sentences into the tensor $\mathbf{X}$, generates $\mathbf{h}_1^{enc}...\mathbf{h}_m^{enc}$, and computes the initial state $\mathbf{h}_0^{dec}$ and initial cell $\mathbf{c}_0^{dec}$ for the Decoder. You can run sanity check by executing `python sanity_check.py 1d` **(5 pts)**

(e) Implement the `decode` function in `nmt_model.py`. This function constructs $\bar{\mathbf{y}}$ and runs the step function over every timestep for the input. You can run sanity check by executing `python sanity_check.py 1e` **(5 pts)**

(f) Implement the `step` function in `nmt_model.py`. This function applies the Decoder's LSTM cell for a single timestep, computing the encoding of the target subword $\mathbf{h}_t^{\text{dec}}$, the attention scores $\mathbf{e}_t$, attention distribution $\alpha_t$, the attention output $\mathbf{a}_t$, and finally the combined output $\mathbf{o}_t$. You can run a non-comprehensive sanity check by executing `python sanity_check.py 1f` **(8 pts)**

(g) Let's train the model! execute the following command:

```
sh run.sh vocab
sh run.sh train
```

Check out the model is running on GPU when training. Training takes within one GPU hour. **(0 pts)**

(h) After training your model, execute the following command to test the model:

```
sh run.sh test
```

Write down the execution time and BLEU score. To get a full credit, BLEU score should be larger than 50. **(3 pts)** **Answer:**

(i) There are a few different methods to generate text from a decoder model such as greedy decoding, beam search, top-k sampling, and top-p sampling. In this code, beam search with a default beam size of 10 is utilized. You can modify the beam size by passing it as an argument in the following way:

```
sh run.sh test <beam-size>
```

Now, perform the decoding with beam size of 1, 3, 5, 10 and 25. Note that beam search with a beam size of 1 is equivalent to greedy decoding. Compare the execution time and performance with different beam sizes. Explain the observed trends as well as the potential reasons for the trends. Discuss distinctions between beam search (beam size greater than 1) and greedy decoding, considering expected and observed differences. **(5 pts)** **Answer:**

(j) **(BONUS)** Conduct additional experiments using various beam sizes to determine the best one. Record your chosen beam size and justify your decision. Research established guidelines for beam size selection (or any rule-of-thumb value for beam size) and contrast your choice or reasoning with these conventions. **(3 pts)** **Answer:**

## 3.4 Written Questions

(a) BLEU score is the most commonly used automatic evaluation metric for NMT systems. It is usually calculated across the entire test set, but here we will consider BLEU defined for a single example.[4] Suppose we have a source sentence $\mathbf{s}$, a set of $k$ reference translations $\mathbf{r}_1, \ldots, \mathbf{r}_k$, and a candidate

[3]LSTMCell: https://pytorch.org/docs/stable/generated/torch.nn.LSTMCell.html. GRUCell:https://pytorch.org/docs/stable/generated/torch.nn.GRUCell.html

[4]This definition of sentence-level BLEU score matches the `sentence_bleu()` function in the `nltk` Python package. Note that the NLTK function is sensitive to capitalization. In this question, all text is lowercased, so capitalization is irrelevant. http://www.nltk.org/api/nltk.translate.html#nltk.translate.bleu_score.sentence_bleu

translation **c**. To compute the BLEU score of **c**, we first compute the *modified n-gram precision* $p_n$ of **c**, for each of $n = 1, 2, 3, 4$, where $n$ is the $n$ in n-gram:

$$p_n = \frac{\displaystyle\sum_{\text{ngram}\in\mathbf{c}} \min\left(\max_{i=1,\ldots,k} \text{Count}_{\mathbf{r}_i}(\text{ngram}),\ \text{Count}_{\mathbf{c}}(\text{ngram})\right)}{\displaystyle\sum_{\text{ngram}\in\mathbf{c}} \text{Count}_{\mathbf{c}}(\text{ngram})} \tag{15}$$

Here, for each of the $n$-grams that appear in the candidate translation **c**, we count the maximum number of times it appears in any one reference translation, capped by the number of times it appears in **c** (this is the numerator). We divide this by the number of $n$-grams in **c** (denominator).

Next, we compute the *brevity penalty* BP. Let $len(c)$ be the length of **c** and let $len(r)$ be the length of the reference translation that is closest to $len(c)$ (in the case of two equally-close reference translation lengths, choose $len(r)$ as the shorter one).

$$BP = \begin{cases} 1 & \text{if } len(c) \geq len(r) \\ \exp\left(1 - \frac{len(r)}{len(c)}\right) & \text{otherwise} \end{cases} \tag{16}$$

Lastly, the BLEU score for candidate **c** with respect to $\mathbf{r}_1, \ldots, \mathbf{r}_k$ is:

$$BLEU = BP \times \exp\left(\sum_{n=1}^{4} \lambda_n \log p_n\right) \tag{17}$$

where $\lambda_1, \lambda_2, \lambda_3, \lambda_4$ are weights that sum to 1. The log here is natural log.

(i) Consider this example.

- Source Sentence **s**: **그때는 뭐 사먹을 것도 엇일 때고 학습장이나 사던지 헤엇던 거 가따**
- Reference Translation $\mathbf{r}_1$: 그때는 뭐 사먹을 것도 없을 때고 학습장이나 사던지 했었던 거 같아
- Reference Translation $\mathbf{r}_2$: 그때는 뭐 사먹을 것도 없을 시절이고 학습장이나 사든지 했었던 거 같다
- NMT Translation $\mathbf{c}_1$: 그때는 뭐 사먹을 것도 없을 때고 학습장이나 사든지 했었던 거 가

Please compute the BLEU scores for $\mathbf{c}_1$. Let $\lambda_i = 0.5$ for $i \in \{1, 2\}$ and $\lambda_i = 0$ for $i \in \{3, 4\}$ (**this means we ignore 3-grams and 4-grams**, i.e., don't compute $p_3$ or $p_4$). When computing BLEU scores, show your working (i.e., show your computed values for $p_1$, $p_2$, $len(c)$, $len(r)$ and $BP$). Note that the BLEU scores can be expressed between 0 and 1 or between 0 and 100. The code is using the 0 to 100 scale while in this question we are using the **0 to 1** scale. **(7 pts)**
**Answer:**

(ii) Due to data availability, NMT systems are often evaluated with respect to only a single reference translation. Please explain (in a few sentences) why this may be problematic. In your explanation, discuss how the BLEU score metric assesses the quality of NMT translations when there are multiple reference transitions versus a single reference translation. **(3 pts)** **Answer:**