IITM-CS7015 : Deep Learning — Given on: March 27, 11:55pm

Programming Assignment #3 — Due on : April 10, 11:55pm IST

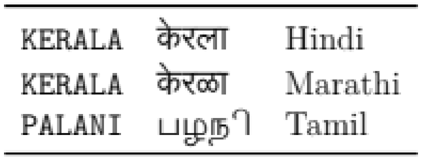
- The aim of this assignment is to train and test a Recurrent Neural Network (LSTM) for text transliteration from English to Hindi.
- We strongly recommend that you work on this assignment in teams of 2. Both the members of the team are expected to work together (and not divide the work) since this assignment is designed with a learning outcome in view.
- Collaborations and discussions with others are strictly prohibited.
- This assignment is going to be time-consuming. **PLEASE START EARLY**. We can not increase the daily submission limit due to technical reasons.
- It will be better if you use **Tensorflow** library (Python) for your implementation (Many of the TAs are comfortable in Tensorflow, for other languages, it will be difficult to get all your doubts cleared). If you are using any other languages, please contact the TAs before you proceed.
- Note that you **Should Not** use any **Higher Level APIs** like keras, estimators, *etc.*. In other words, any API which will support model.train() and model.predict() function is prohibited. You should implement these functions yourself.
- All the models will be tested on same environment on Kaggle. So, you must submit all your results on test data on Kaggle.
- You have to turn in the well documented code along with a report with **Detailed Observations** and inferences of the results electronically in Moodle.
- Typeset your report in Latex code provided (attached). It is necessary to fill '**Checklist**' in the attached sample report. Reports which are not written using Latex will not be accepted.
- The report should be precise and concise. Unnecessary verbosity, (like **Theory about RNNs**) will be penalized.
- You **MUST** run *sanity_check.py* script (attached) on your submission zip file.
- You can find the evaluation pattern (marks distribution) at the last page.
- *You have to check the Moodle discussion forum regularly for updates regarding the assignment.*
- Please ensure that only **ONE** team member submits on Moodle.

1 Task

1.1 Problem Definition

In this assignment you will train and test sequence to sequence networks. We will work with transliteration as an example of a sequence to sequence task.

The goal of transliteration is to write a word in one language using the closest corresponding letters of a different alphabet or language. More formally, the goal of transliteration is to transform a name (a string of characters) in one language (and corresponding script) to the target language (and corresponding script) while ensuring phonemic equivalence (preserving and conforming to the phonology) and conventions of the target language. For instance, the transliteration of the English word NEURAL in Hindi is न्यूरल.
The table below shows some examples of transliterations from English to some Indian languages.



KERALA	केरला	Hindi
KERALA	केरळा	Marathi
PALANI	பழநி	Tamil

If you want to know more, you can read a good survey by [1], though it is now slightly outdated. You can think of this as converting a sequence of characters (k, e, r, a, l, a) from one language to a sequence of characters in another language. You can read more about sequences and seq2seq models here. In this assignment, you have to build a transliteration system from English to Hindi using the NEWS 2012 (Named Entities Workshop) shared task dataset [2].

1.2 Instructions

- Download the NEWS 2012 English-Hindi dataset from kaggle contest's data section. The following is the data split (in number of words):- train: 13122, validation: 997, partial test data: 400.

- Using tensorflow, train a sequence-to-sequence model using the encoder-decoder architecture. The overall structure of the network is as follows:

  (a) INEMBED: A feedforward layer of size $|V_s|$x inembsize, where V_s is the source language vocabulary (i.e. set of characters) and inembsize is the output size of the layer. Use inembsize = 256

  (b) ENCODER: bidirectional LSTM layer with encsize outputs in either direction. encsize = 512

  (c) DECODER: Use a 2-layered decoder. The hidden state of the first decoder should be used to pay attention over the encoder output. Only output of the second decoder should be used in the below softmax layer. Also note that the output of the first decoder should be an input to the second decoder. Use decsize=512. You are **NOT** allowed to use any ready decoder function such as "tf.contrib.seq2seq.Decoder()". You can use the RNNCell (or any variant) but must specifically loop over it to formulate your decoder.

A template pseudocode for the decoder implementation will be similar to the following pseudocode:

```
get encoder_outputs, encoder_state from the encoder layer
for i in range(max_decoding_steps):
    ith_input = function_of(encoder_outputs, decoder_state)
    ith_output, decoder_state = RNNCell(ith_input, decoder_state)
    predicted_char = some_function_of(ith_output)
```

(d) SOFTMAX: softmax layer for classification: decsize inputs and V_t outputs, where V_t is the target language vocabulary (i.e. set of characters).

(e) OUTEMBED: A feedforward layer of size $|Vt|$ x outembsize, where embsize is the output size of the layer. Use outembsize = 256. This layer is used for obtaining the embedding of the output character and feeding it to the input of the decoder for the next time step.

(f) ATTENTION: Incorporate an attention mechanism in your network which will consist of: (i) a single feedforward network whose inputs are the previous decoder state, previous decoder outputs embedding and the annotation vector (encoder output) under consideration. It outputs a single attention score. (ii) a softmax layer over the attention scores from each annotation vector to convert it to a probability value. The attention weights from the softmax layer are used to linearly interpolate the annotation vectors. The resulting context vector will be an input to the decoder along with the decoder output in the previous timestep.

The objective function is to minimize the negative log-likelihood. For inference, you should implement a greedy decoding. Additionally, you can also try to implement beam search. Implementation of beam search is optional (for extra credits).

- Use tanh-nonlinearities for the feedforward as well as LSTM layers.
- Train the network using Adam using the entire training dataset. Use the valid set for validation.
- Use dropout on the output of the encoder and the decoder when training the network.
- Do not use dropout while decoding.
- Use early stopping using the validation set with a patience of 5 epochs.
- Your code should support the following options:
  - --lr (initial learning rate η for gradient descent based algorithms)
  - --batch_size (the batch size to be used - valid values are 1 and multiples of 5)
  - --init (the initialization method to be used - 1 for Xavier, 2 for uniform random)
  - --dropout_prob (the probability of dropping a neuron)
  - --decode_method (0: greedy, 1: beam [default: 0])
  - --beam_width (the beam width, in case beam search is being used)
  - --save_dir (the directory in which the checkpoints will be saved)

  - --epochs (the number of epochs for which model is trained. Epoch: one iteration over training data)
  - --train (training data file to load)
  - --val (validation data file to load)

  You should use the argparse module in python for parsing these parameters.

- Note that this time it is highly important to save and load your model since the final test set will be provided only in the last 2 days. You are expected to have models trained and ready by then. You can load your best models then to get the predictions on the final test data which will have the same format as the current small test dataset named 'partial_test_400.csv'. You can still make submissions on Kaggle currently on this released small test dataset and check your models on it. Final Kaggle scoring will only be based on the accuracies on the final test dataset.

- The primary evaluation metric for the task is Accuracy (exact match). This means that the output is correct only if it matches the reference transliteration exactly. The task is to get an accuracy of 65% on the final test data to be released on April 9th. Meanwhile the partial dataset is intended to guide you in the right directions. It is suggested to train many models and save your good models.

1.3 Kaggle Submission

Important: Note that the test data currently comprises of 400 instances and is only partial test data. On 9th April, we will release the complete dataset. So only on final 2 days (9th and 10th April), you will have access to the complete test dataset. Note that this is the data on which the final scores and marks will be calculated. **Hence everybody is compulsorily required to make a submission to Kaggle on 9th April or 10th April to get their kaggle marks.** Everybody is required to save their best models and use them to transliterate the final test data that gets released on April 9th.

It is suggested that both the team members make separate account on Kaggle. You can form teams for this assignment on the Kaggle assignment page. With this both the team members will be able to make submissions.

You must submit the test_submission.csv files on Kaggle assignment page for evaluation. The evaluation on Kaggle will be done in 2 steps. Till April 8, you are allowed to make 5 submissions on the portal everyday. These will be evaluated on the released partial test data named 'partial_test_400.csv'. On April 9th, we will release the final test set and the final scores start with public and private leaderboards on the full dataset. You will still be allowed 5 submissions per day till April 10. Finally on April 10, you can select 2 of your best submissions to get evaluated in the second step. By default, your best 2 submissions will be taken for evaluation in the second step. After the deadline, your scores in the second step of evaluation will be visible on the Private Leaderboard.

1.4 Report

Prepare a report containing the observations and inferences for the following:

1. A plot of the learning curve showing iterations on the x-axis and negative log likelihood over labels on the y-axis. Make a single plot showing both the training loss and the validation loss. [5 marks]
2. Report the parameter setting which gave the best results and the performance on the test data of the model that performs best on the validation data. (10+15 marks from Kaggle leaderboard)
3. Report a table with validation accuracies with different hyperparameter settings. [15 marks]
4. Write down the dimensions of the input and output at each layer (for example, the input to INEMBED layer is 20 x 50 x 256) [6 marks]
5. Did you try using only a unidirectional LSTM for the encoder? What was the effect? [6 marks total (including implementation/code for unidirectional and bidirectional LSTM)]
6. Explain the attention mechanism you used with equations. [4 marks + 4 for proper implementation/code]
7. What was the effect of using attention mechanism? [2 marks]
8. Plot a visualization of the attention layer weights for a sequence pair. Do you see meaningful character alignments? Do you see one-one, one-many, many-one alignments? Do you see non-contiguous alignments? [7 marks including correct working code]
9. What was the effect of using 2-layered decoder as compared to single decoder? [7 marks including correct code]
10. What was the effect of using dropout? [2 marks]
11. Is the early stopping criterion optimal? Try training for a few epochs beyond the early stopping epoch. Does the validation loss reduce? [3 marks including implementation]
12. Instead of using validation loss as early stopping criterion, you can also try using validation accuracy as stopping criterion. How do the two approaches compare? [2 marks]
13. **[Extra credit]** If you've implemented beam search, how does beam search compare with greedy decoding?
14. **[Extra credit]** What is the effect of beam width on beam search?
15. **[Extra credit]** Please elaborate any additional creative ideas you've implemented

1.5 Submission Instructions:

You need to submit the source code for the assignment. Your code should include one file called *train.py* which should be runnable using the following command:

```
python train.py --lr 0.01 --batch_size 20 --init 1 --save_dir <some_dir>
--dropout_prob <prob_value> --decode_method <value> --beam_width <value>
--epochs 5 --train train.csv --val valid.csv
```

All other supporting files used for generating plots, etc. should also be placed in the zip file. You need a single folder (RollNoTeamMemberA_RollNoTeamMemberB_PA2, *e.g.* CS15D201_CS14B042_PA2) containing the following:

- `train.py`
- `run.sh (containing the best hyperparameters setting)`
- `any other python scripts that you have written`
- `'report.pdf' (in Latex) of the results of your experiments with neatly written answers to the questions mentioned in the above report section`
- `Kaggle_subs/ (folder containing ONLY THE BEST TWO Kaggle submissions). The two filenames in this folder should be 'firstbest.csv' and 'secondbest.csv'.`

The zip should be named as RollNoTeamMemberA_RollNoTeamMemberB_PA3.zip, (*e.g.* CS15D201_CS14B042_PA
Note that zip file and folder name **SHOULD BE SAME**.

Please run sanity_check.py by passing your zip file as command line argument (*e.g.* python sanity_check.py CS15D201_CS14B042_PA3.zip). Only, if this doesn't throws any errors, submit your assignment.

2 References

[1]. Karimi, Sarvnaz, Falk Scholer, and Andrew Turpin. Machine transliteration survey. ACM Computing Surveys, 2011.
[2]. Zhang, Min, Haizhou Li, Ming Liu, and A. Kumaran. Whitepaper of news 2012 shared task on machine transliteration. In Proceedings of the 4th Named Entity Workshop, 2012.

3 Marks Distribution:

Criteria	Marks
Kaggle Score Public	10
Kaggle Score Private	15
Correct implementation of saving and loading a trained model	2
Plot of training loss and validation loss curves	5
Hyperparamater tuning for (learning rate, init, batch_size, dropout_prob)	15
Input and output dimensions of each layer	6
Unidirectional and bidirectional LSTM (code + reporting comparison)	6
Attention mechanism (implementation and report Qs + equations)	10
Attention visualization plot	7
Decoder layers	7
Dropout	2
Early stopping (report questions + implementation)	5
Viva on learnings from the assignment	10
Beam Search (Extra)	10
Additional creative idea (Extra)	10

Table 1: Marks Distribution: Total marks is 100 and extra credits is 20.