IITM-CS7015 : Deep Learning — Given on: April 2, 2:30pm

Programming Assignment #4 — Due on : April 15, 11:55pm IST

- The aim of this assignment is to train and test sequence to sequence networks.
- 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.
- You must use Tensorflow library (Python) for your implementation. You are not allowed to use **high-level APIs** for this task. If you are using any other languages, please contact the TAs before you proceed.
- You have to turn in the well documented code along with a detailed report of the results of the experiment electronically in Moodle. Typeset your report in Latex. Reports which are not written using Latex will not be accepted.
- The report should be precise and concise. Unnecessary verbosity will be penalized.
- You have to check the Moodle discussion forum regularly for updates regarding the assignment.

In this assignment you will train and test sequence to sequence networks. We will work with *Structured Data to Text Descriptions* as an example of a sequence to sequence task. Given a table of facts, you need to generate a description suited for the given table. Figure 1 lists down some examples for this task. You can read more about seq2seq models here.

Instructions

- Download the WEATHERGOV dataset from the following link `https://drive.google.com/file/d/1QlGqN6goVafKZ364Nr6pL0SanJY34viG/view?usp=sharing` . The following is the data split:- train: 25000, validation: 100, test: 1000.
  The dataset folder contains the following files. Please note that the last three files need to be used for hierarchical encoder.
  1. summaries.txt : Ground Truth
  2. *.combined : Structured table of facts as one sequence (to be used for basic encoder)
  3. *.field : List of fields per tables (one table per line)
  4. *.len : Number of tokens per field (one table per line, could be used for dynamic RNN at token level LSTM)
  5. *.proc : First 10 words belong to field 1, Next 10 to field 2 and so on.

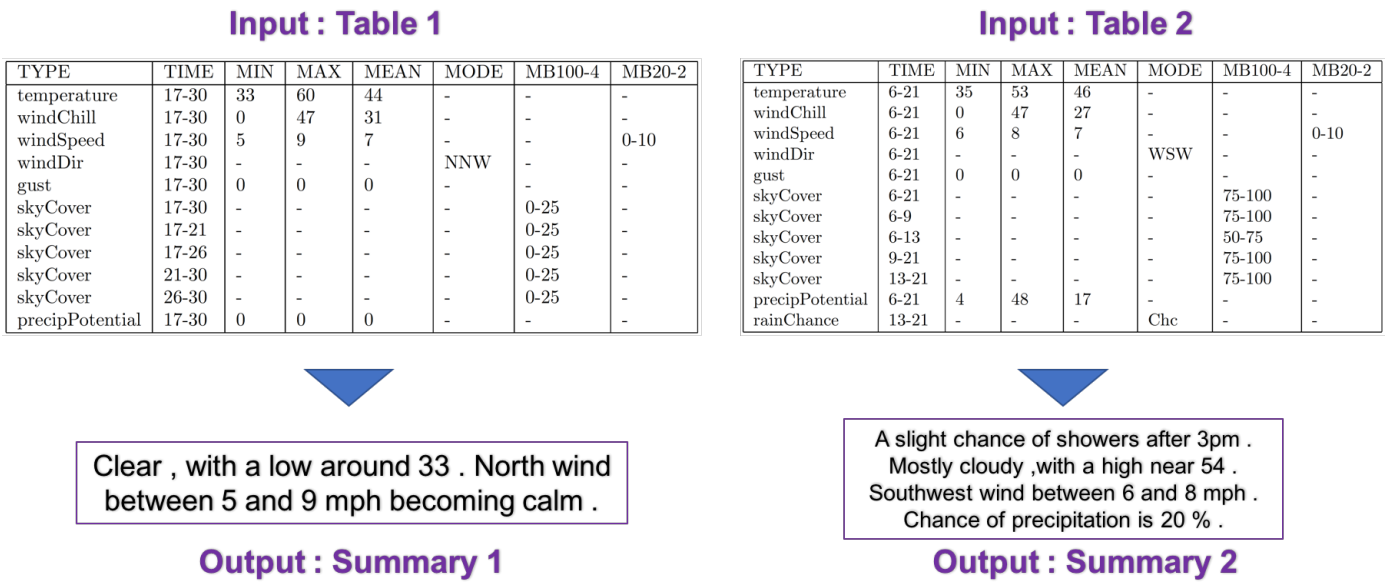

Figure 1: Structured Data to Text

- 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 $|V_s| \times inembsize$, where V_s is the source vocabulary (*i.e.* set of words) and $inembsize$ is the output size of the layer. Use $inembsize = 256$

  (b) **ENCODER**:

  – **BASIC ENCODER** bidirectional LSTM layer with $encsize$ outputs in either direction. $encsize = 512$

  – **HIERARCHICAL ENCODER** The first column in Figure 1 could be considered "Fields" and the corresponding row as the string of "Tokens". The hierarchical encoder will fist encode the tokens using a token level LSTM. Note that the token level bidirectional LSTM should be shared across all the rows. The final state of this LSTM should be then concatenated with the 'field' word embedding and then should be passed through another bidirectional LSTM which basically runs over the first column of the given table.
  For instance, *temperature* is a field, with token string *time 17-30 min 33 max 60 mean 44 ....* G_T is the bidirectional LSTM over these token string which will output the final hidden state H_T which essentially encodes the information about the *temperature* field. This state will be then concatenated with $W_{emb}(temperature)$. These representations for each field will then be passed through another bidirectional LSTM G_F. Pass the final hidden state of this encoder to generate the text descriptions.

  (c) **ATTENTION**: Incorporate an attention mechanism over the **Basic Encoder**

in your network which will consist of: (i) a single feedforward network whose inputs are the previous decoder state, previous decoder output's 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.

(d) **DECODER**: LSTM layer with *decsize* outputs. $decsize = 512$

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

(f) **OUTEMBED**: A feedforward layer of size $|V_t| \times outembsize$, 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.

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 on 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.
- Use different embedding matrix for source tables and the target descriptions.
- The primary evaluation metric for the task is BLEU-4 score. Use the following link `https://drive.google.com/file/d/1GtqfkylKvUfF-NIsDmqhs11Ms_wxtemR/view?usp=sharing` to get the script for the BLEU score. The task is to get a BLEU-4 score of at least 75.00 using basic seq2seq model
  Use command : `python pycocoevalcap/eval.py <file with reference summaries> <file with predicted summaries>`

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>

All other supporting files used for generating plots, etc. should also be placed in the zip file.

- Prepare a report containing the following:
  - Write down the mathematical formulation for both Basic Encoder and Hierarchical Encoder. i.e given the Source Table as a collection of Field words $\{f_1, f_2, ..., f_n\}$ where each field contains $\{w_1^f, w_2^f, ..., w_m^f\}$ tokens. Write the step by steps transformations done to reach the text description Y which is basically a sequence of words $\{y_1, y_2, ..., y_k\}$. **10 marks**
  - 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 for basic encoder-decoder . **5 marks**
  - 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 for hierarchical encoder-decoder . **10 marks**
  - 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 for basic encoder-decoder along with attention mechanism. **10 marks**
  - The performance on the test data of the model that performs best on the validation data. Report some of the test summaries generated from all the three models mentioned above. **10 marks**
  - The parameter setting which gave you the best results. **5 marks**
  - Write down the dimensions of the input and output at each layer (for example, the input to INEMBED layer is $20 \times 50 \times 256$) **5 marks**
  - Did you try using only a unidirectional LSTM for the encoder? What was the effect? **5 marks**
  - Which encoder was better? Basic or Hierarchical Encoder (Without the attention mechanism) ? **5 marks**
  - What was the effect of using attention mechanism? **5 marks**
  - Could you think of a way to implement attention mechanism on top of a hierarchical encoder ? Please mathematically formalize it. No need to implement. **5 marks**
  - Plot a visualization of the attention layer weights for a sequence pair. Do you see meaningful word alignments? Do you see one-one, one-many, many-one alignments? Do you see non-contiguous alignments? **15 marks**

- What is the effect of using dropout? **2.5 marks**
- Is the early stopping criterion optimal? Try training for a few epochs beyond the early stopping epoch. Does the validation loss reduce? **2.5 marks**
- Instead of using validation loss as early stopping criterion, you can also try using validation BLEU as stopping criterion. How do the two approaches compare? **5 marks**
- **Extra Credits** How does beam search compare with greedy decoding? (Implement beam search and report some of the summaries generated through beam search algorithm)
- **Extra Credits** What is the effect of beam width on beam search?