CS7015 Deep Learning
Programming Assignment 4
Sequence to Sequence Network

CS17S008 Nitesh Methani
CS17S013 Pritha Ganguly

April 22, 2018

Contents

List of Figures

1 Structured Data Summarization

Generating natural language description for structured data such as a table is an important task as it helps in easy and fast retrieval of information. Textual data is ever increasing, therefore we need to find some way to condense this data while preserving the information and meaning[10]. In this assignment, we have used the WeatherGov data to create summaries where each weather record data is represented by a vector by it's record type, record time and record values [9]. For example, the chance of rain is represented by *rainChance(time:06:00-21:00, mode:SSE, value:20).*

The given Weather table, T can be viewed as a combination of n field records $\{F_1, \ldots, F_n\}$. Each record consists of a sequence of field values or words $\{w_1, \ldots, w_m\}$ where m is different for different records. The output of the model is the generated summary for the table T which contains p tokens $\{s_1, \ldots, s_p\}$ with s_t being the word generated at time t.

The table to summary generation task is essentially formulated as the inference over a probabilistic model. The goal of the inference is to generate a sequence $s^*_{1:p}$ which maximizes $P(s_{1:p}|F_{1:n})$. This task is accomplished by using a seq2seq model which consists of an encoder and decoder as it's main components.

2 Data Statistics

2.1 Training Data

1. **Source**

Mean Length	89.21
Standard Deviation	14.01
Max Length	122

2. **Target**

Mean Length	30.59
Standard Deviation	13.86
Max Length	88

2.2 Development Data

1. **Source**

Mean Length	89.614
Standard Deviation	13.89
Max Length	122

2. **Target**

Mean Length	30.9
Standard Deviation	13.76
Max Length	86

The dataset was split into 25000 training instances, 1000 instances for development and 3528 instances were used for testing.

3 Variants of RNNs

Recurrent Neural Networks(RNNs) are used to make use of sequential data. For many language modelling tasks such as predicting the next word in a sentence, the information of the previous words which occurred in the sentence is of utmost imoortance. RNNs are called recurrent because they perform the same task for every element of a sequence, with the output being depended on the previous computations. RNNs can be thought of as models with a memory which captures information about the history of what it has been seen so far. There are many variants of RNN cells and they can be categorized in terms of their type, directionality and depth [11].

3.1 Type

There are three types of RNN cells depending on the range with which they capture long term dependencies. The notations used in the equations below are in correspondence to the ones defined in section 6.

3.1.1 Vanilla RNN

The state of a vanilla RNN can be given by:

$$s_t = \sigma(Ux_t + Ws_{t-1} + b)$$

where x_t is the input word at timestep t and s_{t-1} captures all the information the RNN has seen till timestep (t-1). Compactly, a RNN cell can be represented as :

$$s_t = RNN(s_{t-1}, x_t)$$

If the context of the word is far away, RNNs struggle to learn due to vanishing gradient problem. To tackle this issue, LSTMs were proposed.
Note that the above compact representation of RNN is used in the section **??** while writing mathematical formulation of models.

3.1.2 LSTM

Long Short Term Memory(LSTM) cells selectively pass and forget information. The states of a LSTM cell can be represented by :

$$\tilde{s_t} = \sigma(Wh_{t-1} + Ux_t + b)$$

$$s_t = f_t \odot s_{t-1} + i_t \odot \tilde{s_t}$$

$$h_t = o_t \odot \sigma(s_t)$$

where f_t represents the forget gate, i_t represents the input gate and o_t represents the output gate. Compactly, a LSTM cell can be represented as :

$$h_t, s_t = LSTM(h_{t-1}, s_{t-1}, x_t)$$

3.1.3 GRU

Gated Recurrent Units(GRUs) is another variant of LSTMs where there are no explicit forget gate i.e the forget gate and input gates are tied together. The state of a GRU cell can be represented as :

$$\tilde{s_t} = \sigma(W(o_t \odot s_{t1}) + Ux_t + b)$$

$$s_t = i_t \odot s_{t1} + (1 - i_t) \odot \tilde{s_t}$$

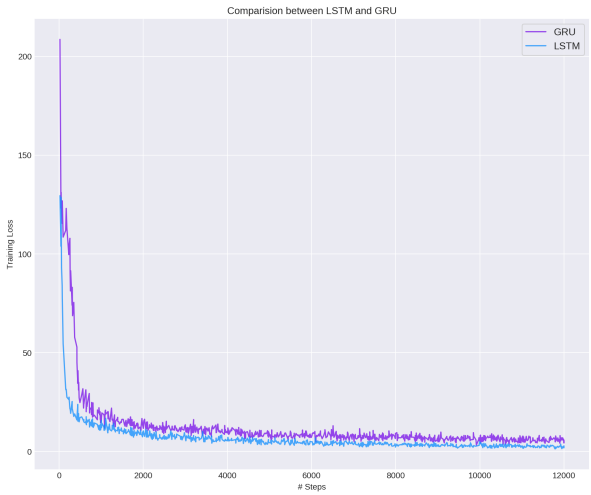
Compactly, a GRU cell can be represented as :

$$s_t = GRU(s_{t-1}, x_t)$$

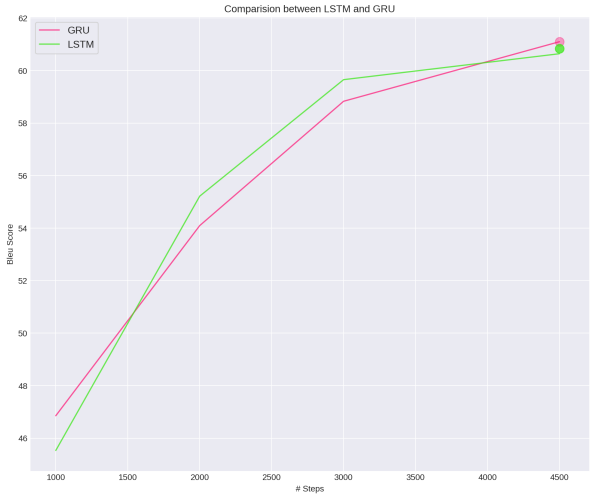
3.1.4 Comparison between GRU and LSTM

- Usually, GRUs train faster and perform better than LSTMs on less training data especially for language modeling tasks.
- GRUs are simpler and thus easier to modify. [15]
- LSTMs remember longer sequences than GRUs and outperform them in tasks requiring modeling long-distance relations.
- GRU exposes the complete memory unlike LSTM. [14]

  But in this assignment we are doing data summarization where we need to remember longer sequences, we will be using LSTM cells in our encoder decoder model.



(a) Training Loss



(b) Bleu Score

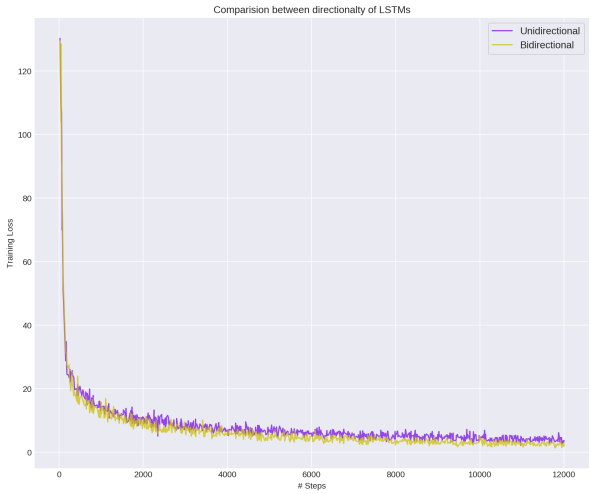
Figure 1: Effect of choice of RNN cell type on seq2seq model

For the structured data summarization task, we can see that the LSTM cell performs better than the GRU cell. Note that the circle over the line in bleu score plot represents the best bleu score.

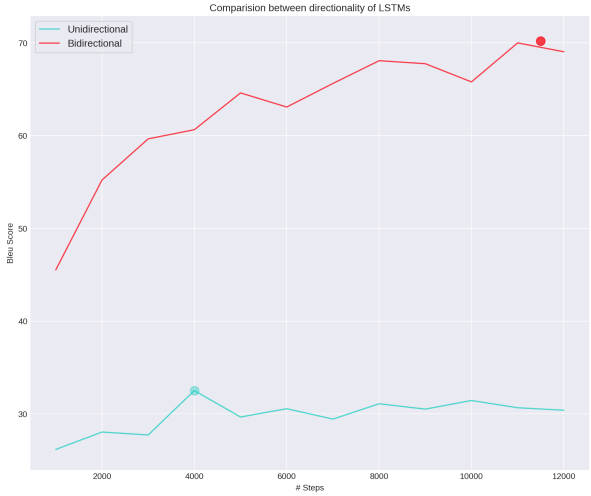
3.2 Directionality

We usually talk about directionality of LSTM cells.

- Unidirectional LSTM only preserves information of the past as it's inputs at time step t depends on what it has seen till time step (t-1).
- Bidirectional LSTMs runs the inputs in two ways to preserve information from the past as well as the future so that both the left and the right context of the data is captured at any point in time.



(a) Training Loss



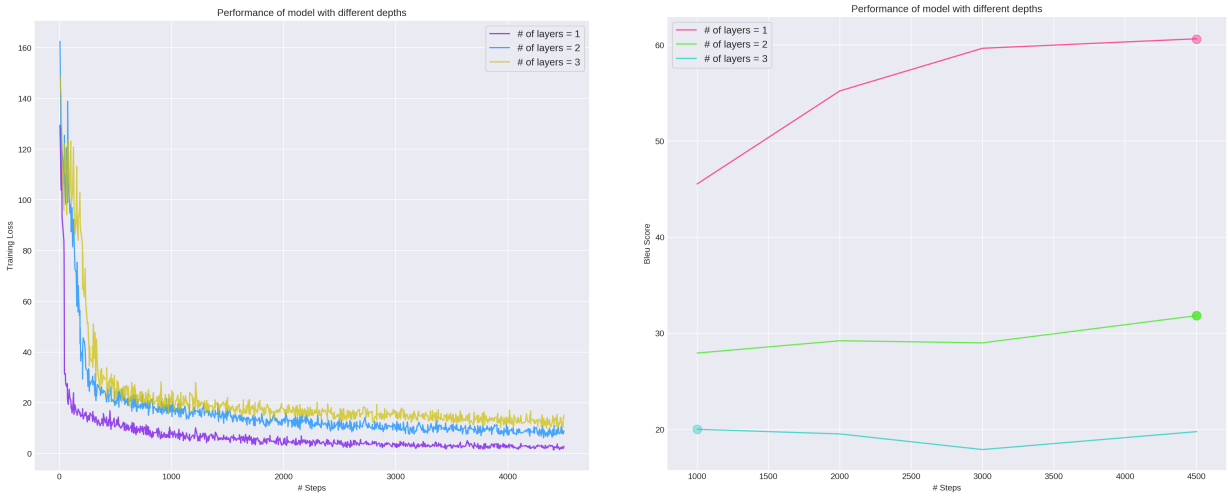
(b) Bleu Score

Figure 2: Effect of directionality of LSTMs on seq2seq model

For the data summarization task, bidirectional LSTMs perform better than the unidirectional ones.

3.3 Depth

Deep RNN networks are similar to shallow RNNs, only that there are multiple layers per time step. In practice this gives a higher learning capacity at the cost of a lot of training data. Any type of the above discussed RNN cells can be used to implement deep networks.



(a) Training Loss

(b) Bleu Score

Figure 3: Effect of depth in terms of layers on seq2seq model

For the data summarization task, single layer bidirectional LSTMs perform better than multi-layer LSTMs.

4 Word Embedding

In a seq2seq model, the encoder encodes the table while the decoder decodes the representation to generate a summary. In this context, we cannot pass the input table as text data. We create embedding, therefore we first create a vocabulary list containing all the words we want the model to be able to use or read. The model inputs will have to be tensors containing the IDs of the words in the sequence.
There are three symbols, that we need our vocabulary to contain. They are :

- $< s >$: Start of sentence token; this is the input to the first time step of the decoder to let the decoder know when to start generating output.
- $< \backslash s >$: End of sentence token; it allows us to tell the decoder where a sentence ends, and it allows the decoder to indicate the same thing in its outputs as well.
- $< unk >$: Unknown symbol; while building the vocabulary, the words which are very less frequent are often replaced with this symbol to improve resource efficiency.

The source and target word embeddings can be learnt via word2vec model or it can be learnt while training the seq2seq model.

5 Encoder

The encoder encodes the input sequence into a thought vector. Each word x_t, where t is the time step from the input sequence is associated to a vector via an embedding look-up table. Then, the LSTM is run over this sequence of vectors and last hidden state generated by the LSTM is stored: this will be our encoder representation h_T.

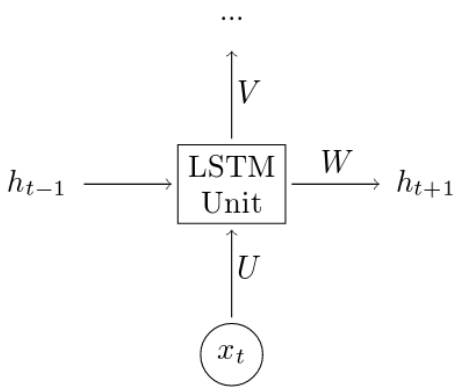


Figure 4: Encoder Cell

6 Decoder

Now that we have a vector h_T that captures the meaning of the input sequence, it can be used to generate the target sequence word by word. This vector h_T, and start of sentence token is fed to another LSTM cell(as it's initial hidden state) which marks the beginning of the decoder circuit. The LSTM will takes hidden state s_{t-1} and the output generated at time step t-1 $e(y_{t-1})$ as input and outputs a probability vector p_t over the next word, etc. The decoding stops when the predicted word is a special end of sentence token. The decoder aims at modelling the distribution of the next word conditioned on the beginning of the sentence.

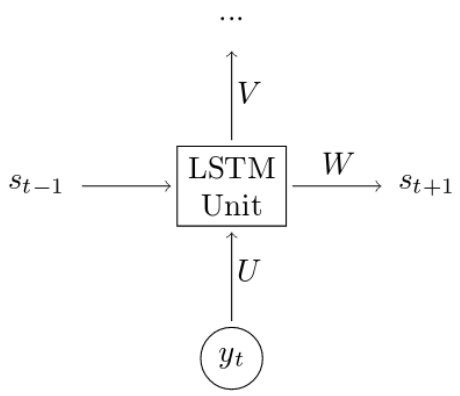


Figure 5: Decoder Cell

7 Attention Mechanism

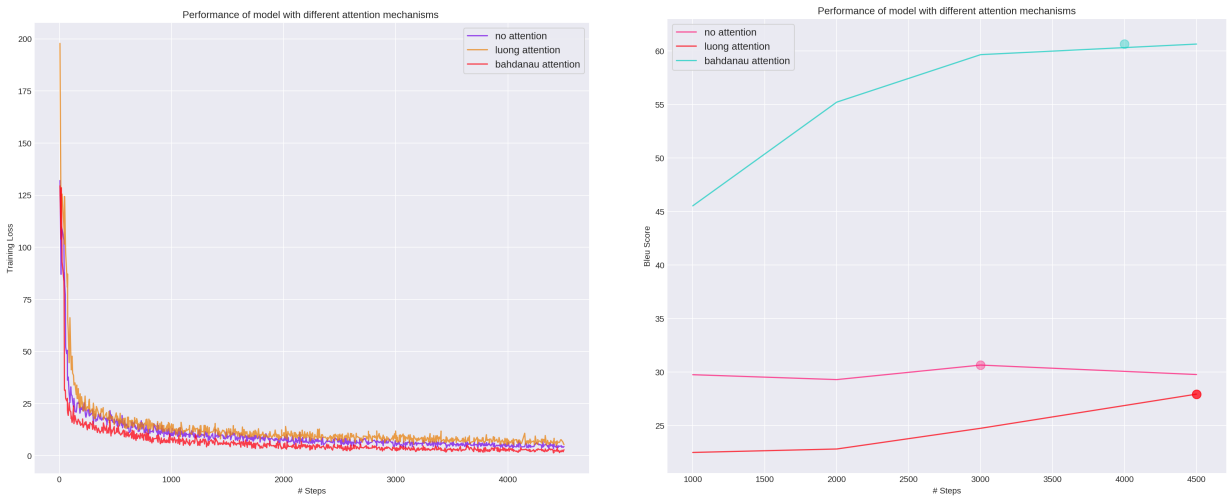
The basic encoder-decoder model fails to scale up. The main bottleneck is the fixed sized thought vector, therefore it is not able to capture all the relevant information of the input sequence as the model size increases. At each generation step, only a part of the input is relevant. This is where attention comes it. It helps the model decide which part of the input encoding to focus on at each generation step to generate novel words.
There are two popular methods of attention mechanism:

- Luong Attention mechanism : Here the alignment at time step t is computed by using hidden state at time step t, h_t and all source hidden states. To integrate the context vector, it creates an independent RNN-like structure to take the concatenation of c_t and h_t as input and predicts y_t.
- Bahdanau Attention mechanism : Here the alignment at time step t is computed by using hidden state at time step t-1, i.e h_{t-1}. To integrate the context vector c_t, it chooses to

concatenate it with the hidden state h_{t-1} as the new hidden state which is fed to the next step to generate h_t as well as predict y_{t+1}

The plots shows the effect of attention vs no attention mechanism on training loss and bleu score(on developmental data). We can clearly see that attention mechanisms give a much better performance. Among Luong and Bahdanau mechanisms, the latter performs better on bidirectional LSTMs.

7.1 Attention Mechanism



(a) Training Loss

(b) Bleu Score

Figure 6: Effect of attention mechanism on seq2seq model

7.2 Visualizations

- **Source Table**
  temperature time 17-30 min 48 mean 53 max 67 windChill time 17-30 min 0 mean 17 max 50 windSpeed time 17-30 min 2 mean 2 max 6 mode-bucket-0-20-2 0-10 windDir time 17-30 mode ESE gust time 17-30 min 0 mean 0 max 0 skyCover time 17-30 mode-bucket-0-100-4 50-75 skyCover time 17-21 mode-bucket-0-100-4 50-75 skyCover time 17-26 mode-bucket-0-100-4 50-75 skyCover time 21-30 mode-bucket-0-100-4 50-75 skyCover time 26-30 mode-bucket-0-100-4 50-75 precipPotential time 17-30 min 6 mean 12 max 14

- **Generated Summary (after 12000 steps)**
  Mostly cloudy , with a low around 47 . Light east wind .

- **Expected Summary**
  Mostly cloudy , with a low around 47 . Light east wind .

In the plot shown below, the y-axis represents the Source Table(size 77 tokens) whereas the x-axis shows the Generated Summary(size 13 tokens) for both 11000 and 12000 time steps. The plot shows the attention which the network gives on each of the table token values to generate the summaries.

(a) After 11000 Steps

(b) After 12000 Steps

Figure 7: Visualization of Attention weights

8 Mathematical Formulation

The notations used in the formulations are in correspondence to the ones defined in section 5 and 6.

8.1 Basic Encoder without attention mechanism

- **Encoder**
  The Encoder takes input x_t and encodes that to a thought vector [1].

$$h_0 \leftarrow random\ initialization$$
$$h_t = RNN(x_t, h_{t-1})$$

  The last hidden state of the encoder (h_T) gives the encoded thought vector, which is fed as the initial state of the decoder.

- **Decoder**
  The Decoder, which is also a RNN, takes encoded thought vector and previous states as inputs and gives the output.

$$s_0 \leftarrow h_T$$
$$s_t = RNN(e(\hat{y}_{t-1}), s_{t-1}) ....(1)$$
$$s_t = RNN([e(\hat{y}_{t-1}), y_t], s_{t-1}) ....(2)$$

  Equation (1) takes the output of the previous step $e(\hat{y}_{t-1})$ and s_{t-1} while generating the current state s_t. This is implemented as part of GreedyEmbeddingHelper() in Tensorflow.

Whereas equation (2), along with the above two inputs, also takes the target outputs as inputs in the decoder cell. This is implemeneted as part of TrainingHelper() in Tensorflow. Usually, equation 2 is used for the initial few epochs and then onwards, equation 1 takes over.

8.2 Basic Encoder with attention mechanism

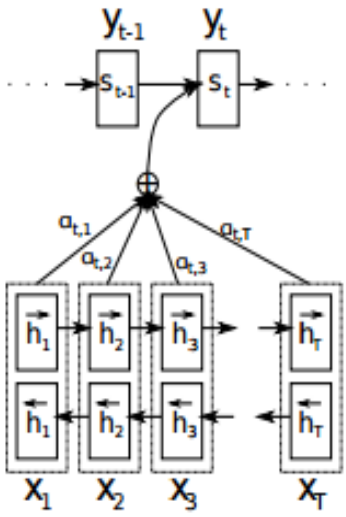


Figure 8: Model with Attentions

- **Encoder**
  The Encoder takes input x_t and encodes that to a thought vector.

$$h_0 \leftarrow random\ initialization$$
$$h_t = RNN(x_t, h_{t-1})$$

  The last hidden state of the encoder (h_T) gives the encoded thought vector, which is fed as the initial state of the decoder.

- **Decoder**
  The Decoder, takes encoded thought vector weighed by the attention parameter $\alpha's$ and previous states as inputs and gives the output. We have to learn these attention parameters using the following equations. Attention distribution is represented by α_{jt} and the final thought/context vector is c_t.

$$s_0 \leftarrow h_T$$
$$e_{jt} = V_{att}^T tanh(W_{att} h_j + U_{att} s_{t-1})$$
$$\alpha_{jt} = softmax(e_{jt})$$
$$c_t = \sum_{j=1}^{T} \alpha_{jt} h_j$$
$$s_t = RNN([e(\hat{y}_{t-1}), c_t], s_{t-1})$$
$$l_t = softmax(V s_t + b)$$

  l_t is the final output at time step t.

8.3 Hierarchical Encoder without attention mechanism

In this task of table summarization, we can apply a two level hierarchical RNN. The first level or token level RNN encodes the tokens of each individual fields and gives us a representation. The

second level or field level RNN takes these token representations along with the embedding of each field to give a single representation of the sequences of tokens. Note that the notations used are consistent with the ones used in section 1.

- **Token Level Encoder**
  The representation of the j^{th} token of the i^{th} field is represented by :

$$h_{ij}^1 = RNN(h_{ij-1}^1, w_{ij})$$
$$s_i = h_{iT_i}^1$$

  where T is the length of field i.
  Each of the s_i is the token level representation of each field F_i. These sequences of token are fed to the Field level encoder along with the respective embedding of it's field (f_i).

- **Field Level Encoder** The representation of each field is given by :

$$h_i^2 = RNN(h_{i-1}^2, \{s_i, f_i\})$$
$$s = h_n^2$$

  where n is the number of fields.

- **Decoder**
  The Decoder is same as the one discussed in the previous subsection 8.1. Now the initial vector fed to the decoder circuit will be s.

8.4 Hierarchical Encoder with attention mechanism

To apply attention mechanism over an hierarchical encoder, we need attention at two levels. First we need to attend to important (most informative) tokens in each field. Then we need to attend to important (most informative) fields in the table to generate summaries.

- **Token Level Encoder**
  The mathematical formulations are given by :

$$h_{ij} = RNN(h_{ij-1}, w_{ij})$$
$$u_{ij} = tanh(W_w h_{ij} + b_w)$$

  where W_w, b_w represents weight matrix and bias term respectively with respect to the tokens of fields.

$$\alpha_{ij} = \frac{exp(u_{ij}^T u_w)}{\sum_t exp(u_{it}^T u_w}$$
$$s_i = \sum_j \alpha_{ij} h_j$$

- **Field Level Encoder**
  The mathematical formulations are given by :

$$h_i = RNN(h_{i-1}, \{s_i, f_i\})$$
$$u_i = tanh(W_f h_i + b_f)$$

  where W_f, b_f represents weight matrix and bias term respectively with respect to the fields.

$$\alpha_i = \frac{exp(u_i^T u_f)}{\sum_i exp(u_i^T u_f}$$
$$s = \sum_i \alpha_i h_i$$

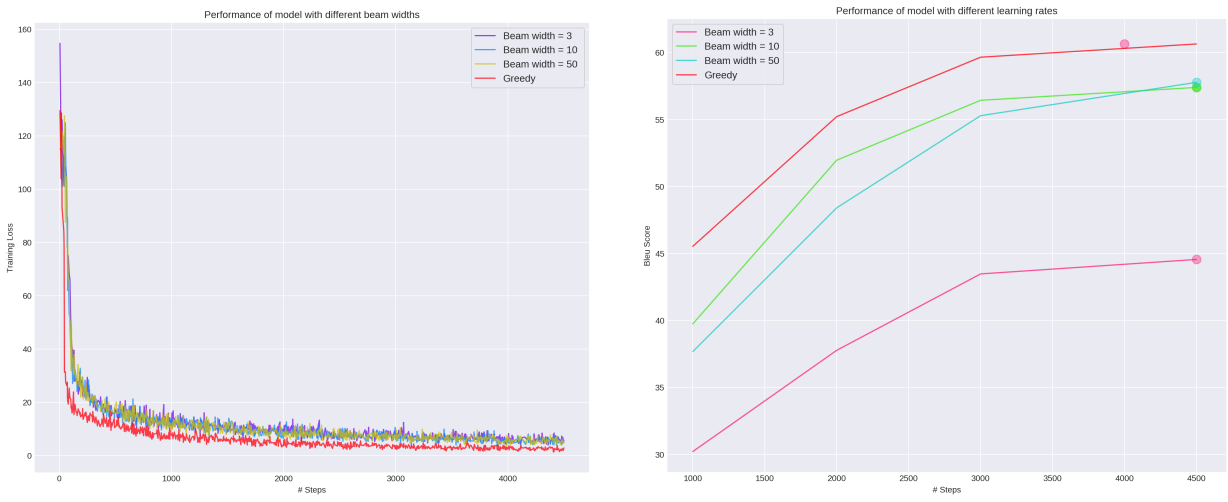
- **Decoder**
  The Decoder is same as the one discussed in the previous subsection 8.2

9 Hyperparameter Space

1. Number of Units : 256 | 512 | 1024
2. Embedding Size : 128 | 256 | 512
3. Encoder Cell Type : uni | bi
4. Number of Layers : same | diff
5. Encoder Depth: 1 | 2 | 3
6. Decoder Depth: 1 | 2 | 3
7. Optimizer: adam | sgd
8. Learning Rate: 0.01 | 0.001 | 0.0001
9. Training Steps: 4500
10. Share Vocabulary: True | False
11. Unit Type: lstm | gru
12. Dropout: 0.2 | 0.5
13. Batch Size: 16 | 32 | 128
14. Metrics: bleu | accuracy
15. Beam Width: 3 | 10 | 50

10 Beam Search

Instead of greedily choosing the most likely next step as the sequence is constructed, the beam search expands all possible next steps and keeps the k most likely, where k is a user-specified parameter and controls the number of beams or parallel searches through the sequence of probabilities [13]. The search doesn't need to start with random states, but with the k most likely words as the first step in the sequence.
Common beam width values are 1 for a greedy search and values of 3 or 10 for common benchmark problems in machine translation. Larger beam widths usually result in better performance of a model as the multiple candidate sequences increase the likelihood of better matching a target sequence. This increased performance results in a decrease in decoding speed. Although in the experiments shown below, greedy search performs better both with respect to training loss and dev bleu score.



(a) Training Loss

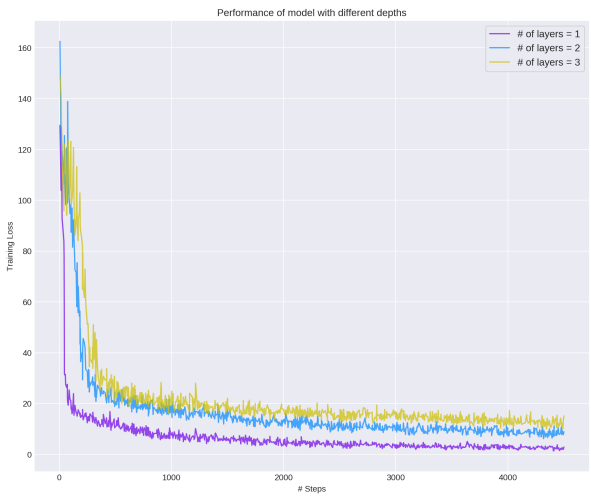
(b) Bleu Score

Figure 9: Effect of beam widths on seq2seq model

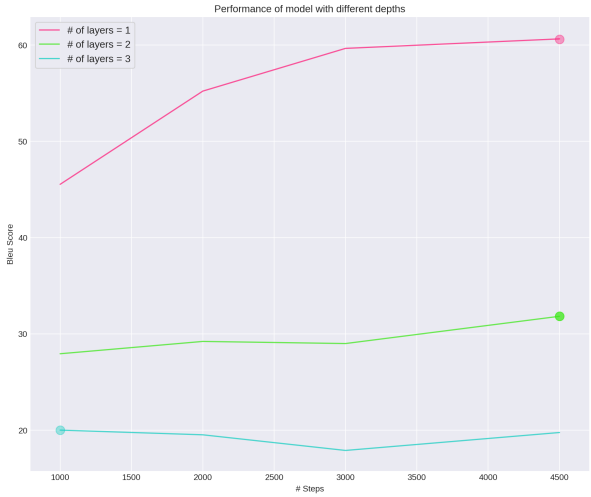
11 Experiments

We have performed different experiments (as discussed below) in order to tune the hyper-parameter space. We have plotted the performance of the seq2seq network in terms of training loss and bleu score on development data, with different parameter settings.

11.1 Depth of network



(a) Training Loss

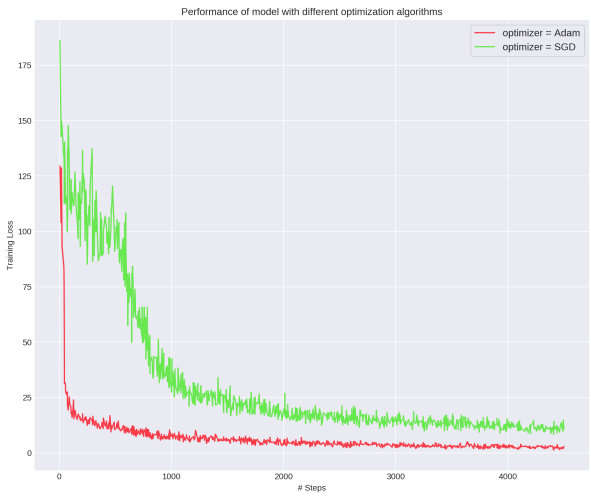


(b) Bleu Score

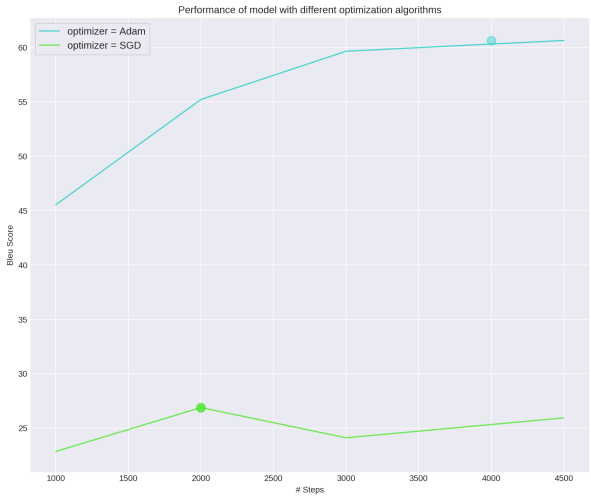
Figure 10: Effect of different depths of seq2seq model

Though in general there's a usual belief that multilayered LSTM models perform better than single layered LSTM model, but in the context of summary generation from tables, the plots show that the latter performs much better.

11.2 Adam vs SGD



(a) Training Loss

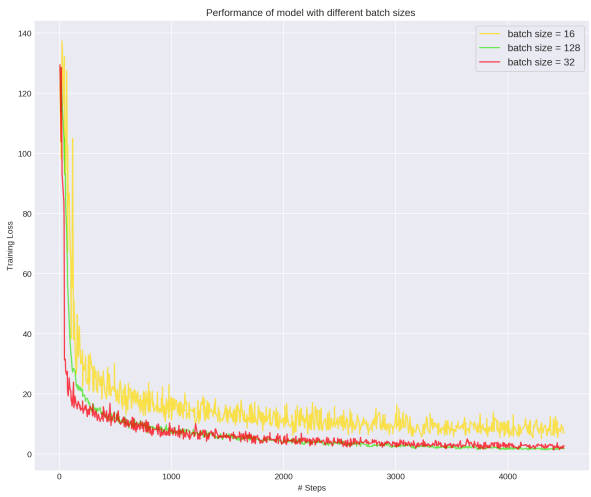


(b) Bleu Score

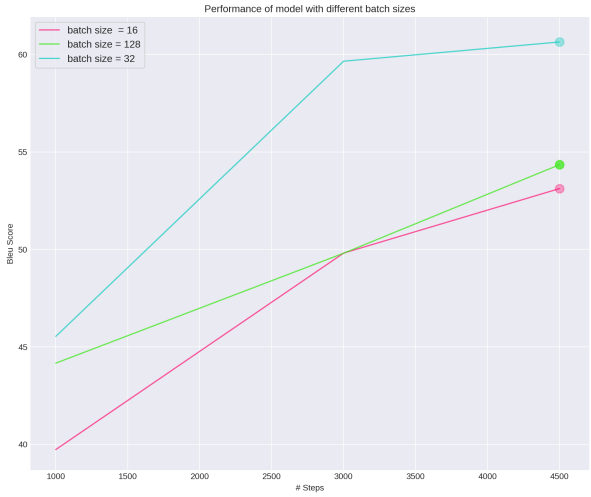
Figure 11: Effect of different optimizers on seq2seq model

As expected, while experimenting with optimization algorithms, ADAM performed much better than stochastic gradient descent algorithm.

11.3 Batch Size



(a) Training Loss

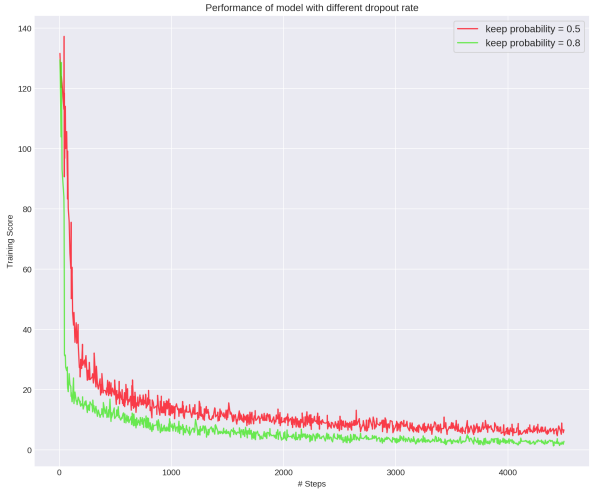


(b) Bleu Score

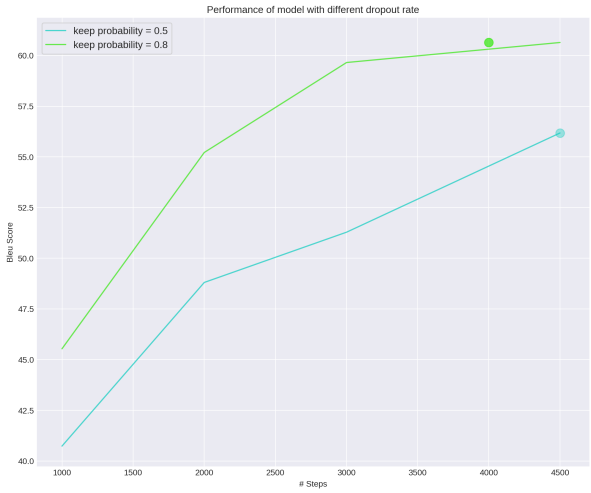
Figure 12: Effect of batch size while training on seq2seq model

The usual notion is smaller the batch size, the more updates the weight matrices get in the gradient descent algorithm. But, experiments show that batch size of 32 and 128 perform significantly better than the batch size of 16 with respect to the training loss. In terms of bleu score on development data, batch size of 32 receives the highest bleu score.

11.4 Dropout



(a) Training Loss

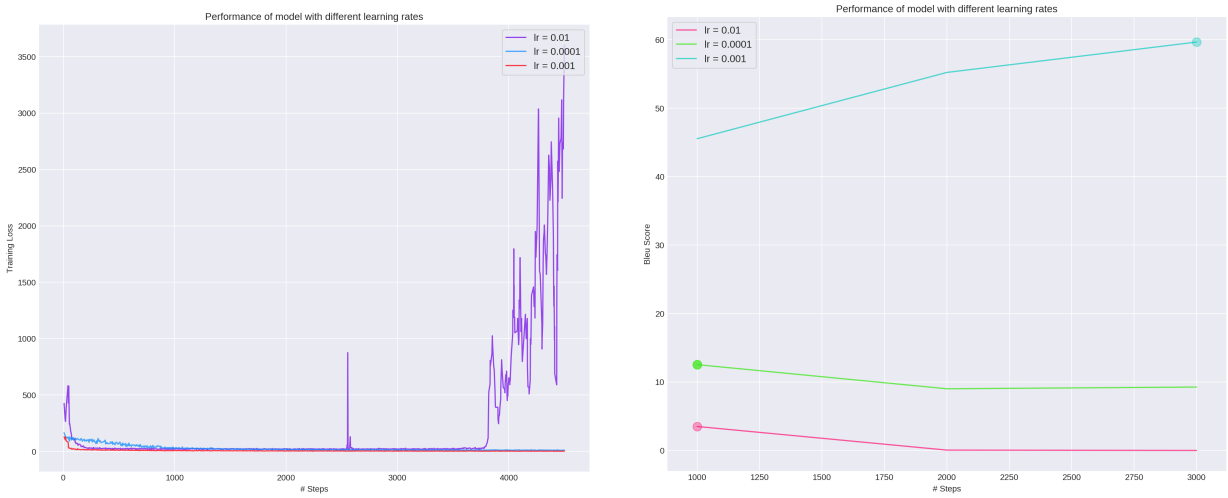


(b) Bleu Score

Figure 13: Effect of dropout probability on seq2seq model

An issue with LSTMs is that they can easily overfit training data, reducing their predictive skill. Dropout is a regularization method where input and recurrent connections to LSTM units are probabilistically excluded from activation and weight updates while training a network. This has the effect of reducing overfitting and improving model performance. As we can see when we drop 20% of the connections, the performance is better as compared to dropping 50% connections as in the latter case, important connections may be lost.

11.5 Learning Rate

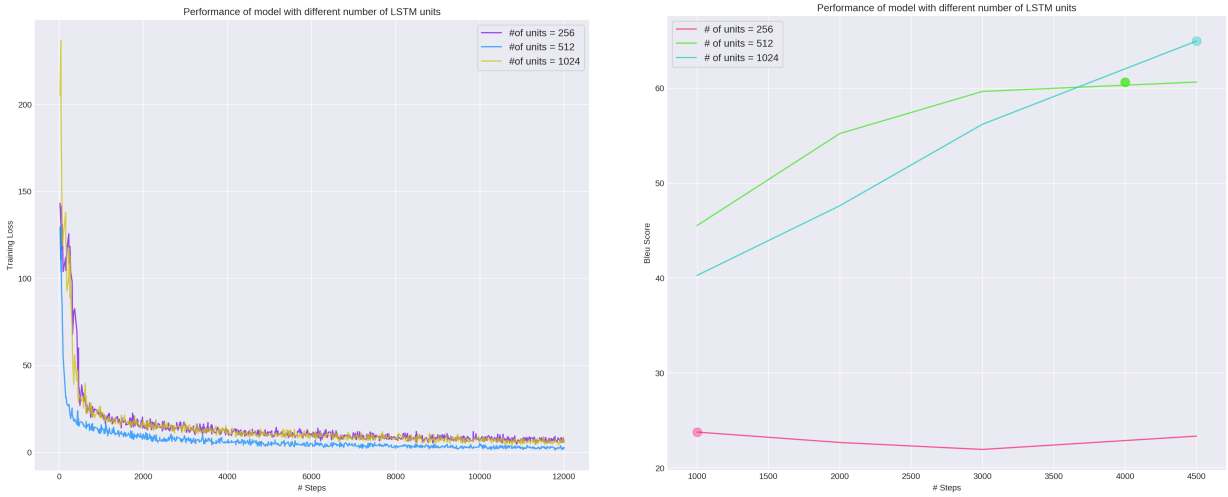


(a) Training Loss (b) Bleu Score

Figure 14: Effect of learning rates on seq2seq model

After choosing ADAM as the optimization algorithm, we experiment with the learning rate. Learning rates of 0.001 and 0.0001 perform almost similarly whereas a learning rate of 0.01 undergoes serious oscillations.

11.6 Number of units



(a) Training Loss (b) Bleu Score

Figure 15: Effect of number of LSTM units on seq2seq model

The number of hidden units is a direct representation of the learning capacity of a neural network - it reflects the number of learned parameters. Using more units makes it more likely to perfectly memorize the complete training set (although it will take longer, and you run the risk of overfitting). 512 units seems to be decent enough to not overfit the data.

11.7 Early Stopping

Early stopping is a form of regularization used to avoid over-fitting when training a learner with an iterative method. Generally validation loss is used for this purpose. But as we know that Bleu score is a metric for evaluating a generated sentence to a reference sentence, we can also use Bleu score as an early stopping criteria.
In the plot below we have compared both of these criteria.

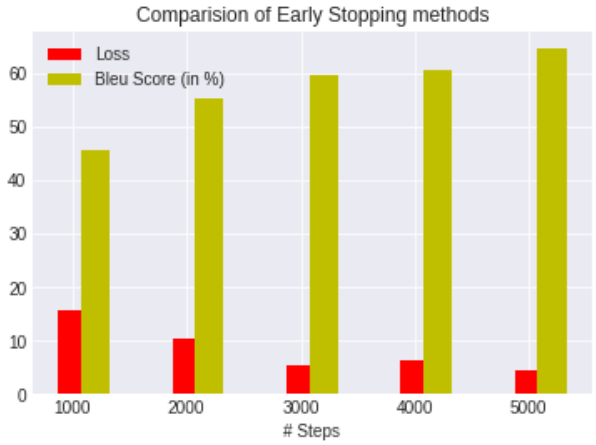


Figure 16: Comparison of Early stopping methods

As we can see from the above plot that although BLEU has significant advantages, there is no guarantee that an increase in BLEU score is an indicator of improved translation quality.

12 Best Model

1. Number of Units : 512
2. Embedding Size : 256
3. Encoder Cell Type : bi
4. Number of Layers : same
5. Encoder Depth: 1
6. Decoder Depth: 1
7. Optimizer: adam
8. Learning Rate: 0.001
9. Training Steps: 12000
10. Share Vocabulary: False
11. Unit Type: lstm
12. Dropout: 0.2
13. Batch Size: 32
14. Metrics: bleu
15. Beam Width: 1 (GreedySearch)

13 Number of Parameters

Note that:

- Table vocab size : 192
- Summary vocab size : 392
- Embedding Size : 256
- Max table length : 122 tokens
- Max summary length : 88 tokens
- Batch Size : 32
- Output Units : 512

The dimensions of the input and output at each layer of basic seq2seq model are as follows :

- Pre-processing : Encoder Embedding
  - INPUT : $(192, 1)$
  - OUTPUT : $(192, 256)$
- Layer 1 : Encoder
  - INPUT : $(32, 122, 256)$
  - STATE : $(32, 512)$
  - OUTPUT : $(32, 122, 512)$
- Pre-processing : Decoder Embedding
  - INPUT : $(392, 1)$
  - OUTPUT : $(392, 256)$
- Layer 2 : Decoder
  - INPUT : $(32, 88, 256)$
  - STATE : $(32, 512)$
  - OUTPUT : $(32, 88, 392)$

14 Observations

We compared the summaries generated by different models. Few of them are reported below :

1. **Unidirectional v/s Bidirectional**

   (a) Training Summaries

Summaries
Mostly sunny , with a high near 53 . North wind between 7 and 10 mph .
Mostly cloudy , with a low around 46 . South wind between 3 and 6 mph .
Mostly cloudy , with a low around 50 . East wind between 5 and 10 mph , with gusts as high as 15 mph .

Table 1: Unidirectional Model

Summaries
Mostly sunny , with a high near 53 . Southeast wind between 7 and 9 mph .
A 50 percent chance of showers . Mostly cloudy , with a low around 47 . South wind around 5 mph .
Mostly cloudy , with a low around 49 . Calm wind becoming south southeast around 5 mph .

Table 2: Bidirectional Model

Summaries
Mostly sunny , with a high near 53 . Southeast wind between 7 and 9 mph .
A 50 percent chance of showers . Cloudy , with a low around 47 . East wind around 5 mph .
Cloudy , with a low around 49 . Calm wind becoming south southeast around 5 mph .

Table 3: Expected summaries

   (b) Development Summaries

Summaries
Sunny , with a high near 46 . West wind between 7 and 9 mph .
A 30 percent chance of snow showers , mainly after 11pm . Mostly cloudy , with a low around 29 . West wind between 5 and 10 mph .
A 50 percent chance of showers , mainly before 10pm . Mostly cloudy , with a low around 36 . West wind around 5 mph becoming south .

Table 4: Unidirectional Model

Summaries
Sunny , with a high near 46 . West wind between 6 and 9 mph .
Partly cloudy , with a low around 29 . West southwest wind 5 to 10 mph becoming north northwest .
A chance of rain showers before 3am , then a chance of rain and snow showers . Patchy fog before 3am . Otherwise , mostly cloudy , with a low around 36 . Breezy , with a west wind between 14 and 20 mph . Chance of precipitation is 50

Table 5: Bidirectional Model

Summaries
Sunny , with a high near 46 . West wind between 6 and 9 mph .
Partly cloudy , with a low around 29 . Northwest wind between 5 and 10 mph .
A chance of rain showers before 3am , then a chance of rain and snow showers . Patchy fog before 3am . Otherwise , mostly cloudy , with a low around 35 . Breezy , with a west wind between 14 and 20 mph . Chance of precipitation is 50 % . Little or no snow accumulation expected .

Table 6: Expected summaries

2. **No Attention v/s Attention**

(a) Training Summaries

Summaries
Mostly sunny , with a high near 53 . West wind between 8 and 15 mph , with gusts as high as 23 mph .
A 50 percent chance of showers after 10pm . Mostly cloudy , with a low around 47 . West wind between 5 and 10 mph , with gusts as high as 20 mph .
Mostly cloudy , with a low around 50 . South wind between 5 and 15 mph .

Table 7: Model without attention

Summaries
Mostly sunny , with a high near 53 . Southeast wind between 7 and 9 mph .
A 50 percent chance of showers . Mostly cloudy , with a low around 47 . South wind around 5 mph .
Mostly cloudy , with a low around 49 . Calm wind becoming south southeast around 5 mph .

Table 8: Model with Attention

Summaries
Mostly sunny , with a high near 53 . Southeast wind between 7 and 9 mph .
A 50 percent chance of showers . Cloudy , with a low around 47 . East wind around 5 mph .
Cloudy , with a low around 49 . Calm wind becoming south southeast around 5 mph .

Table 9: Expected summaries

(b) Development Summaries

Summaries
A chance of rain or drizzle before 9am . Mostly cloudy , with a high near 45 . West wind between 5 and 11 mph . Chance of precipitation is 30 % .
A 10 percent chance of snow showers before 10pm . Patchy fog after 11pm . Otherwise , mostly cloudy , with a low around 31 . West wind between 5 and 9 mph .
Mostly cloudy , with a low around 35 . Southwest wind between 11 and 16 mph , with gusts as high as 23 mph .

Table 10: Model without attention

Summaries
Sunny , with a high near 46 . West wind between 6 and 9 mph .
Partly cloudy , with a low around 29 . West southwest wind 5 to 10 mph becoming north northwest .
A chance of rain showers before 3am , then a chance of rain and snow showers . Patchy fog before 3am . Otherwise , mostly cloudy , with a low around 36 . Breezy , with a west wind between 14 and 20 mph . Chance of precipitation is 50

Table 11: Model with Attention

Summaries
Sunny , with a high near 46 . West wind between 6 and 9 mph .
Partly cloudy , with a low around 29 . Northwest wind between 5 and 10 mph .
A chance of rain showers before 3am , then a chance of rain and snow showers . Patchy fog before 3am . Otherwise , mostly cloudy , with a low around 35 . Breezy , with a west wind between 14 and 20 mph . Chance of precipitation is 50 % . Little or no snow accumulation expected .

Table 12: Expected summaries

3. **Beam Search v/s Greedy Search**
   Here we used a beam width of 10.

   (a) Training Summaries

Summaries
Mostly sunny , with a high near 53 . Southeast wind between 7 and 9 mph .
A 50 percent chance of showers . Mostly cloudy , with a low around 47 . South wind around 5 mph .
Mostly cloudy , with a low around 49 . Calm wind becoming south southeast around 5 mph .

Table 13: Model with Greedy Search Decoder

Summaries
Mostly sunny , with a high near 53 . Southeast wind between 7 and 9 mph .
Mostly cloudy , with a low around 47 . East northeast wind around 5 mph becoming east .
Patchy fog after 1am . Otherwise , mostly cloudy , with a low around 49 . Calm wind becoming south around 5 mph .

Table 14: Model with Beam Search Decoder

Summaries
Mostly sunny , with a high near 53 . Southeast wind between 7 and 9 mph .
A 50 percent chance of showers . Cloudy , with a low around 47 . East wind around 5 mph .
Cloudy , with a low around 49 . Calm wind becoming south southeast around 5 mph .

Table 15: Expected summaries

(b) Development Summaries

Summaries
Sunny , with a high near 46 . West wind between 6 and 9 mph .
Partly cloudy , with a low around 29 . West southwest wind 5 to 10 mph becoming north northwest .
A chance of rain showers before 3am , then a chance of rain and snow showers . Patchy fog before 3am . Otherwise , mostly cloudy , with a low around 36 . Breezy , with a west wind between 14 and 20 mph . Chance of precipitation is 50

Table 16: Model with Greedy Search Decoder

Summaries
A chance of rain or drizzle after 9am . Mostly cloudy , with a high near 46 . South wind between 3 and 9 mph . Chance of precipitation is 30 % .
A 20 percent chance of rain after midnight . Increasing clouds , with a low around 29 . Southwest wind between 5 and 10 mph .
A chance of rain showers before 3am , then a chance of rain and snow showers . Patchy fog before 3am . Otherwise , mostly cloudy , with a low around 35 . Breezy , with a west wind between 14 and 20 mph . Chance of precipitation is 50

Table 17: Model with Beam Search Decoder

Summaries
Sunny , with a high near 46 . West wind between 6 and 9 mph .
Partly cloudy , with a low around 29 . Northwest wind between 5 and 10 mph .
A chance of rain showers before 3am , then a chance of rain and snow showers . Patchy fog before 3am . Otherwise , mostly cloudy , with a low around 35 . Breezy , with a west wind between 14 and 20 mph . Chance of precipitation is 50 % . Little or no snow accumulation expected .

Table 18: Expected summaries

15 Conclusion

In this assignment we tried out different hyperparameter settings with different variants of RNNs. We found that GRU is computationally easier than LSTM since it has only 2 gates and it's performance is on par with LSTM but LSTM outperforms GRU when we have to remember longer sequences. Bidirectional LSTMs produce much better summaries than their unidirectional counterpart as they take both the past and future context into account. Also, attention mechanism significantly improves performance as the model learns to pay attention to only important part of the table to generate the summaries.
During the course of this assignment we were exposed to Neural Net frameworks like Tensorflow. We gained practical experience with the training and managed to produce reasonably good results.

References

[1] Mitesh Sir's slides, *course slides*

[2] Colah Blog *http://colah.github.io/posts/2015-08-Understanding-LSTMs/*

[3] Tensorflow *https://github.com/tensorflow/nmt*

[4] CharRNN *https://r2rt.com/recurrent-neural-networks-in-tensorflow-ii.html*

[5] Diagrams *https://github.com/dennybritz/rnn-tutorial-gru-lstm/tree/master/latex*

[6] Hvass-Labs *https://github.com/Hvass-Labs/TensorFlow-Tutorials*

[7] Hvass-Labs *https://github.com/lmthang/thesis*

[8] Karpathy's blog *http://karpathy.github.io/2015/05/21/rnn-effectiveness/*

[9] Table-to-text Generation by Structure-aware Seq2seq Learning
https://arxiv.org/pdf/1711.09724.pdf

[10] Abstractive Text Summarization -Soumye Singhal & Prof. Arnab Bhattacharya
http://home.iitk.ac.in/ soumye/cs498a/pres.pdf

[11] Introduction to RNNs
http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/

[12] NMT
https://arxiv.org/pdf/1409.0473v6.pdf

[13] Beam Search
https://machinelearningmastery.com/beam-search-decoder-natural-language-processing/

[14] GRU vs LSTM
https://arxiv.org/pdf/1412.3555v1.pdf

[15] Empirical GRU
http://proceedings.mlr.press/v37/jozefowicz15.pdf

[16] Dimensions
https://www.quora.com/In-LSTM-how-do-you-figure-out-what-size-the-weights-are-supposed-to-be