Programming Assignment 4

Subrajit Makur (CS17m046) & Amar Vashishth (CS17m052)

April 22, 2018

CheckBox

- Network Structure
  - ☑ InEmbed layer
  - ☑ Basic Encoder (bidirectional)
  - ☐ *Hierarchical Encoder
  - ☑ Attention Mechanism
  - ☑ Decoder
  - ☑ Softmax
  - ☑ OutEmbed
  - ☑ Greedy Decoding
  - ☑ Beam Search
  - ☑ BLEU Score Evaluation Metric
- Report Requirements
  - ☑ Mathematical Formulation for Basic Encoder
  - ☑ Training, Validation Loss Vs Number of Epochs, w.r.t. Basic Encoder
  - ☑ Training and Validation loss Vs Number of Epochs, with Attention Mechanism + Basic Encoder
  - ☑ Report of some Test Summaries
  - ☑ Best Parameters Settings
  - ☑ Dimensions of I/O at each layers
  - ☑ Effect Bi Directional LSTM Encoder
  - ☐ *Without Attention, Basic Vs Hierarchical Encoder
  - ☑ Effect of using Attention Mechanism
  - ☑ *In Theory, way to implement Attention over Hierarchical Encoder
  - ☑ Visulization of Attention layer weights
  - ☑ Effect of using Dropouts
  - ☑ Comment on the use of Early Stopping
  - ☑ Comment on the use of Validation BELU instead of Early Stopping
  - ☑ *Beam Search Vs Greedy Encoding
  - ☑ *Effect of Beam Width on Beam Search

Mathematical Formulation for Basic Encoder

For a Recurrent Neural network which has a single hidden layer, with input denoted as X, weights of hidden layer denoted as w_h and the weights of output layer denoted as w_y, weights of recurrent computation denoted as w_r, hidden representation denoted as h and the output denoted as y,

$$h^t = \sigma(W_h\ X + W_r\ h^{t-1})$$
$$y = \sigma(W_y\ h^t)$$

When such a RNN is bidirectional the equation set will change to:

$$h_1^t = \sigma(W_{h1}\ X + W_{r1}\ h_1^{t-1})$$
$$h_2^t = \sigma(W_{h2}\ X + W_{r2}\ h_2^{t+1})$$
$$y = \sigma(W_{y1}\ h_1^t + W_{y2}\ h_2^t)$$

subscript 1 and 2 denotes the variables associated with hidden layer 1 and hidden layer 2.

When we use **LSTM**:
states: values that are used to offer the information for output.

- input data: it is denoted as x.
- hidden state: values of previous hidden layer. This is the same as traditional RNN. It is denoted as h.
- input state: values that are (linear) combination of hidden state and input of current time step. It is denoted as i, and we have:

$$i^t = \sigma(W_{ix}\ x^t + W_{ih}\ h^{t-1})$$

- internal state: Values that serve as "memory". It is denoted as m

gates: values that are used to decide the information flow of states.

- input gate: it decides whether input state enters internal state. It is denoted as g, and we have:

$$g^t = \sigma(W_{gi}\ i^t)$$

- forget gate: it decides whether internal state forgets the previous internal state. It is denoted as f, and we have:

$$f^t = \sigma(W_{fi}\ i^t)$$

- output gate: it decides whether internal state passes its value to output and hidden state of next time step. It is denoted as o and we have:

$$o^t = \sigma(W_{oi}\ i^t)$$

Finally, considering how gates decide the information flow of states, we have the last two equations to complete the formulation of LSTM:

$$m^t = g^t \odot i^t + f^t\ m^{t-1}$$
$$h^t = o^t \odot m^t$$

where, $\odot$ denotes element-wise product.

Attention Mechanism:
To mathematically formalize this mechanism. We use r to denote the encoded representation (there is a total of M regions of representation), use h to denote hidden states of LSTM cell. Then, the attention module can generate the unscaled weights for i^{th} region of the encoded representation as:

$$\beta_i^t = f(h^{t-1},\ r,\ \{\alpha_j^{t-1}\}_{j=1}^M)$$

where, α_j^{t-1} is the attention weights computed at the previous time step, and can be computed at current time step as a simple softmax function:

$$\alpha_i^t = \frac{\exp(\beta_i^t)}{\sum_j^M \exp(\beta_j^t)}$$

Therefore, we can further use the weights α to reweight the representation r for prediction. Here we use soft attention, which is simply defined as:

$$r^t = \sum_j^M \alpha_j^t c_j$$

Training, Validation Loss Vs Number of Epochs, w.r.t. Basic Encoder

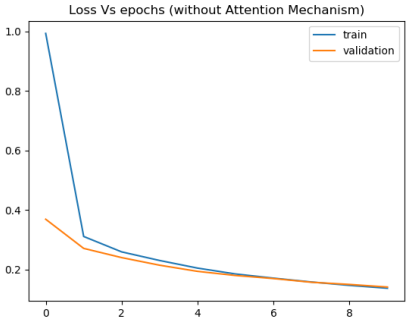

Figure 1: Loss Vs Epochs, Without Attention Mechanism

Training and Validation loss Vs Number of Epochs, with Attention Mechanism + Basic Encoder

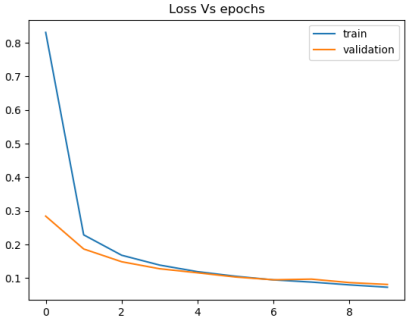

Figure 2: Loss Vs Epochs, With Attention Mechanism

Report of some Test Summaries

- With Attention:
  - Showers likely , mainly after 1pm . Mostly cloudy , with a high near 63 . Breezy , with a south wind between 15 and 20 mph , with gusts as high as 30 mph . Chance of precipitation is 60 % . New rainfall amounts between a tenth and quarter of an inch , except higher amounts possible in thunderstorms .
  - A slight chance of showers , then occasional showers and thunderstorms after 1pm . High near 67 . South wind between 10 and 20 mph , with gusts as high as 30 mph . Chance of precipitation is 80 % . New rainfall amounts between a quarter and half of an inch possible .
  - Showers and thunderstorms likely . Cloudy , with a low around 56 . South wind between 10 and 20 mph , with gusts as high as 30 mph . Chance of precipitation is 70 % .
  - Showers likely , mainly after 1pm . Cloudy , with a high near 53 . South wind between 7 and 11 mph . Chance of precipitation is 60 % . New rainfall amounts between a tenth and quarter of an inch possible .
- Without Attention:
  - Mostly cloudy , with a low around 64 . East wind around 15 mph .
  - Mostly clear , with a low around 62 . East wind around 14 mph , with gusts as high as 20 mph .
  - Rain and possibly a thunderstorm , mainly after midnight . Mostly cloudy , with a low around 35 . Southwest wind between 5 and 10 mph . Chance of precipitation is 70 % . New rainfall amounts of less than a half inch possible .
  - A slight chance of showers . Mostly cloudy , with a low around 59 . North wind around 5 mph becoming east . Chance of precipitation is 20 % .

Best Parameters Settings

Parameter	Optimal Value
Embedding Dimensions	256
RNN Cell Variant	LSTM
Encoder	Bidirectional
Attention Dimension	512
Attention Type	Bahdanau or Luong
Optimizer	Adam
Learning Rate	0.001
Beam Width	5
Dropout probability	0.2
Batch Size	100

Dimensions of I/O at each layers

Layer	Dimension
InEmbed	$100 \times \lvert V_s \rvert \times 256$
Basic Encoder	$100 \times \lvert V_s \rvert \times 512$
Decoder Input	$100 \times$ max_output_sentence_length
Decoder Embedding Input	$100 \times$ max_output_sentence_length $\times 256$
Decoder Embedding Output	$100 \times$ max_output_sentence_length $\times 512$
Softmax	$100 \times \lvert V_d \rvert$

Effect of Bi Directional LSTM Encoder

A Bidirectional RNN is a combination of two RNNs — one runs forward from left-to-right and the other one runs backwards from right-to-left. It encodes past and future information. It could be represented as:

$$h_t = \left[\overleftarrow{h_t},\ \overrightarrow{h_t}\right]$$

Since they are using both past and future information, they are able to understand the context of the input statement better than a unidirectional lstm. Their usage, improved BLEU score.

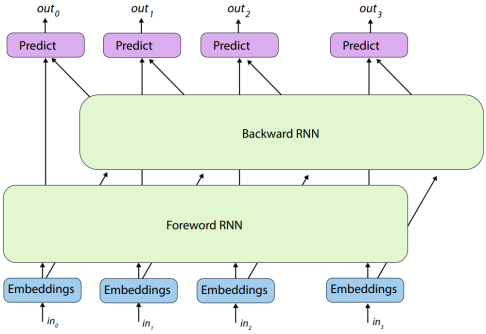

Figure 3: Bi-Directional RNN

Effect of using Attention Mechanism

Attention based models outperform the vanilla encoder-decoder model. Without Attention model yielded a BLEU score of 0.35 with Epoch 100
With Attention model yielded a BLEU score of 0.55 with Epoch 25

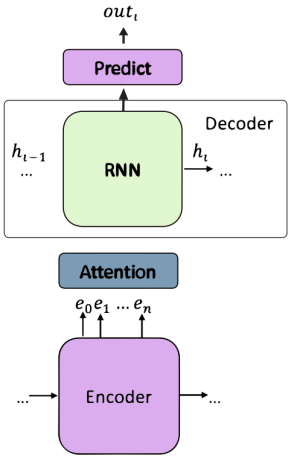

Figure 4: RNN using Attention Mechanism

*In Theory, way to implement Attention over Hierarchical Encoder

We could try to identify key sentences for the summary. Given that the source document is big enough. We can use two bi-directional RNN at source text:

- one at Word Level
- another at Sentence Level

Word level attention is the weighted by corresponding Sentence level attention:

$$P^a(j) = \frac{P_w^a(j)\ P_s^a\Big(s(j)\Big)}{\sum_{k=1}^{N^d} P_w^a(k)\ P_s^a\Big(s(k)\Big)}$$

Visualization of Attention layer weights

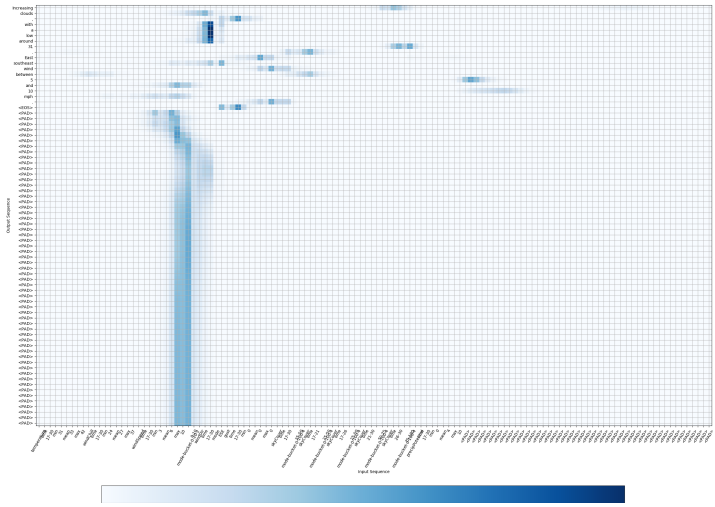

Figure 5: Visualization of Attention Layer Weights

Effect of using Dropouts

The idea is very simple: randomly dropping out some of the units while training. More formally: on each training case, each hidden unit is randomly omitted from the network with a probability of p(in our case, $p = 0.2$).

Dropout can be seen as an efficient way to perform model averaging across a large number of different neural networks, where overfitting can be avoided with much less cost of computation.

Comment on the use of Early Stopping

There are three phases to training:
Phase I, also called as the Error domination phase, where the network hardly learns anything and is still very biased. During this phase the approximation error is further and further reduced. Although, during this phase the complexity error increases. This is induced by the increasing variance of the network model as the possible magnitude and diversity of the weights grows.

If we train long enough the error will be dominated by the complexity error, which we also call **Phase III**.

Therefore, there is a phase during training, when the approximation and complexity (or: bias and variance) components of the error compete but none of them dominates **Phase II**.

Early stopping detects when Phase II ends and when the dominance of variance part begins. It provides a technique for computing a stopping point based on complexity considerations without using a separate validation set.

Comment on the use of Validation BLEU instead of Early Stopping

A true evaluation of a trained model can be done by inference and a BLEU score (not validation). In fact, we have seen instances with very good validation perplexity which obtain bad BLEU score.

We felt that Bleu is not the best metric here. Indeed, that could be just an impression. It cannot reflect the actual quality of NMT.

Use of Validation BLEU instead of Early stopping is a bad idea.

*Beam Search Vs Greedy Encoding

Beam Search:
Choosing the top k most likely target words and then feeding them all into the next decoder input. So at each time-step t the decoder gets k different possible inputs.

It then computes the top k most likely target words for each of these different inputs. Among these, it keeps only the top-k out of k^2 and rejects the rest. This process continues. This ensures that each target word gets a fair shot at generating the summary.

Greedy Encoding:
It is the most natural way. Feeds the best token to the next step; Feeding to the next step, the most likely predicted word of the previous step.

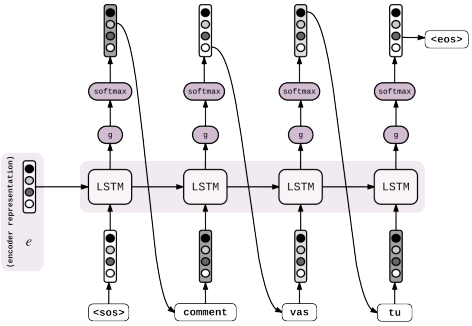

Figure 6: Greedy Encoding

*Effect of Beam Width on Beam Search

We tried increasing beam width gradually from 0 to 10 and found that as we increase beam width, corresponding BLEU score also increases.