Assignment-2 : Cooler Machine Translation – Igpay Atinlay

CSE 291(G) - Wintern 2019

General Instructions to be kept in mind:

1. You are allowed to use a library of your choice for this task, consisting of but not limited to, TensorFlow, PyTorch and Keras.
2. Starter code has been provided in an iPython notebook. Please try to keep your edits limited to this file **only**.
3. Obviously there are multiple ways to code this out and the starter code is just one of the many different ways, you are free to make edits to the function definitions and/or code to suit yourself.
4. There are multiple hints provided in the starter code so it's recommended that you go through the notebook first before you make any edits. This would also be helpful in understanding the overall flow of the code.
5. The submission consists of the edited iPython Notebook and a report (1-2 pages) consisting of answers to questions and the details/insights obtained from your experiments.
6. This assignment has to be done **individually** abiding by the Academic Integrity Policy which has already been relayed to you by the instructor and is included on the syllabus.

1 Introduction

Hope you guys had fun with the last assignment and were successfully able to train the Pig-Latin model. In this assignment we will build upon the previous assignment and apply attention mechanism to assist our model to perform better. *You can use your code from the previous assignment wherever you wish.*

2 Attention?

Attention is a way of providing the model with more flexibility on what part of the input it wants to focus on while generating the output at each time step.

More specifically, we want that at time-step t of the decoder pass, the attention network learns a weighted function over the encoder annotations. $\alpha(t)$ denotes a vector of attention weights, having size equal to the length of the input sequence (if the batch size is 1) for the decoder time-step t. The attention weights have the following properties :

1. The attention weights are non-negative and normalized i.e., $\alpha^i(t) >= 0$ and $\sum_i \alpha^i(t) = 1$
2. They are a function of the previous decoder hidden state $h_{dec}(t)$ and the vector of the hidden states of the encoder h_{enc}.
$$\alpha^i(t) = f(h_{dec}(t-1); h_{enc}(i-1)) \tag{1}$$
3. The function f can take many different forms but we will implement a mini neural-network consisting of two layers, one hidden and one output layer, for this function f. The neural network would take in a concatenated vector (as shown in equation 1) and output a single number $\alpha^i(t)$.

Question 1. Implement the Attention Class and fill out the forward pass. You will find a need to define the function f which would take in inputs of size $hidden\_size * 2$ and output a number. Finally, apply softmax over the attention outputs obtained.

TIP: You do not need for loops for this. Think of matrix multiplications and some reshaping and squeezing/un-squeezing.

3 Modified Decoder

Once you have these normalized attention weights given by your Attention class, you form a *context* vector out of these. A context vector is simply a weighted vector of the encoder hidden annotations, the weight given to each encoder hidden annotations is determined by the attention weight.

More specifically, $c(t)$ denotes a vector,

$$c(t) = \sum_{i=1}^{T} \alpha^i(t) h_{enc}(i-1) \tag{2}$$

This context vector is then concatenated with the input at the current time step of the decoder and we repeat the usual process. So, instead of the input to the RNN being just $x(t)$, the new input becomes $[x(t); c(t)]$

Question 2. Fill out the code for the Decoder forward pass but this time you will have an extra input in the form of encoder hidden states and three additional steps to perform compared to the last Decoder you implemented:

1. Embed the inputs
2. **Find the attention weights using an object of the attention class**
3. **Form the context vector**
4. **Concat the context vector and embedding vector**
5. Pass it through the RNN
6. Get outputs of size equal to the vocablury size

4 Visualize Attention

Attention is beautiful in the sense that you can usually gain insights into the learning of the model. The code for this has already been provided to you and for each epoch you will have attention color maps for the test word that you have defined in the params.

Question 3. Play around with your trained model and visualize the attention weights for different words, specifically, vowels, consonants, compound-words (words separated by a dash) and report your interesting findings along with the color maps.

5 Experiments

You are required to run experiments for different kinds of implementations of the function f. Be creative!

Question 6. Report the hyper-parameters and the loss plots for the best attention function that you can find and show us the color maps.