KUSHAGRA SETH
UCSC Id: 2005986
kuseth@ucsc.edu

Homework Assignment-3
(Spring 2023)
NLP-203

Professor: Ian Lane
TA: Geetanjali Rakshit

1 Problem-1

1. **Defining the Goal Function**
   For every time unit $t = 0, \ldots, T$, define a binary variable x_{ij}^t for each potential shift from tag i to tag j. The goal function that needs optimization can be put as:

$$\max \sum_{t=0}^{T} \sum_{i,j} c_{ij} x_{ij}^t$$

   where c_{ij} denotes the score (or cost) linked to the shift from tag i to tag j.

2. **Asserting One Transition per Time Unit Constraints**
   For every time unit t, only one transition should occur. This can be written as:

$$\sum_{i,j} x_{ij}^t = 1 \quad \forall t$$

   As we need to express this with inequality constraints, it can be split into two inequalities:

$$\sum_{i,j} x_{ij}^t \leq 1 \quad \forall t$$

$$\sum_{i,j} x_{ij}^t \geq 1 \quad \forall t$$

3. **Valid Sequence Formation through Transitions**
   To guarantee that a transition from i to j precedes a transition from j to k, we apply the following conditions:

$$\sum_{i} x_{ij}^t = \sum_{k} x_{jk}^{t+1} \quad \forall j, t$$

   Again, this should be stated as two inequalities:

$$\sum_{i} x_{ij}^t \leq \sum_{k} x_{jk}^{t+1} \quad \forall j, t$$

$$\sum_{i} x_{ij}^t \geq \sum_{k} x_{jk}^{t+1} \quad \forall j, t$$

4. **Implementing Constraints for BIO Tagging**
   In BIO tagging, "I" (Inside) can't come after "O" (Outside). This constraint can be presented as, if we have a shift from tag "O" to tag "I" at time t, then there should be no transition at time $t + 1$. This is expressed as:

$$x_{OI}^t + \sum_{k} x_{Ik}^{t+1} \leq 1 \quad \forall t$$

   This assures that if "I" succeeds "O", then "I" can't be followed by any tag in the next step. Note that the x_{OI}^t and x_{Ik}^{t+1} are binary variables, so when x_{OI}^t equals 1 (implying "I" comes after "O"), the second term has to be 0, thereby enforcing the necessary constraint.

2 Problem-2

For projectivity, we introduce a new binary variable y_{ijk} for each triple of words (w_i, w_j, w_k) such that $y_{ijk} = 1$ if the parent of w_k is in the range $[i, j]$, and 0 otherwise. We can then express the projectivity constraint as:

$$y_{ijk} \geq x_{ij} + x_{ik} - 1 \quad \forall i < j < k \tag{1}$$

$$y_{ijk} \leq x_{ij} \quad \forall i < j < k \tag{2}$$

$$y_{ijk} \leq x_{ik} \quad \forall i < j < k \tag{3}$$

3 Problem-3

Baseline Model Architecture:

```
Seq2Seq(
  (encoder): Encoder(
    (embedding): Embedding(515, 64)
    (rnn): GRU(64, 128, bidirectional=True)
    (fc): Linear(in_features=256, out_features=128, bias=True)
    (dropout): Dropout(p=0.5, inplace=False)
  )
  (decoder): Decoder(
    (attention): Attention(
      (attn): Linear(in_features=384, out_features=128, bias=True)
      (v): Linear(in_features=128, out_features=1, bias=False)
    )
    (embedding): Embedding(515, 64)
    (rnn): GRU(320, 128)
    (fc_out): Linear(in_features=448, out_features=515, bias=True)
    (dropout): Dropout(p=0.5, inplace=False)
  )
)
The model has 701,251 trainable parameters
```

Figure 1: Baseline Model

The Encoder class belongs to the Seq2Seq model and its main function is to convert the input sentence into a context vector. It accomplishes this with an embedding layer and a Gated Recurrent Unit (GRU) layer. The GRU layer is bidirectional, as suggested by its bidirectional parameter, and dropout is used to combat overfitting.
The Attention class is a part of the Seq2Seq model that allows the model to concentrate on different segments of the encoder's outputs for each step of the decoder's outputs. It utilizes the previous hidden state of the decoder and all the forward and backward hidden states from the encoder to yield the attention vector.
The Decoder class forms the other part of the Seq2Seq model and its main objective is to generate a sequence of words from an input or a collection of inputs. An embedding layer and a GRU layer are used here as well, but there is also a linear layer for final prediction. The Attention class, defined earlier, is used to implement the attention mechanism.
The Seq2Seq class is a comprehensive representation of the sequence-to-sequence process. It includes

the implementations of both the Encoder and the Decoder and manages the overarching process of receiving source data, processing it through the encoder and decoder, and producing the results.
The Seq2Seq class's forward method initiates the mask, obtains the input and output from the encoder, and then decodes the input step by step within a loop. Each time step's decoder output is saved in the outputs tensor. The method of teacher forcing is employed here, which is an effective and quick way to train recurrent neural network models by using the ground truth from a previous time step as input.

The *Rouge Scores* for the Summaries on the Test set are mentioned below:

Rouge-1	**Rouge-2**	**Rouge-L**
0.1976	0.0334	0.1879

Table 1: Baseline Scores