به نام خدا

دانشگاه تهران

دانشکدگان فنی

دانشکده مهندسی برق و کامپیوتر

درس پردازش زبان طبیعی

پاسخ تمرین شماره سه

نام و نام خانوادگی: امیرحسین شاه قلی

شماره دانشجویی: ۸۱۰۱۹۹۴۴۱

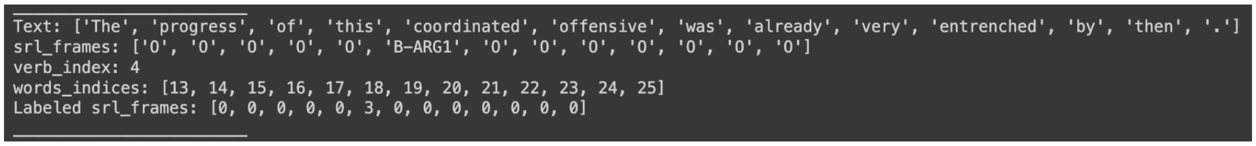
اردیبهشت ماه ۱۴۰۳

Table of Contents

Answer of the question one

Answer of the first part:

Second sentence part:



```
Text: ['The', 'progress', 'of', 'this', 'coordinated', 'offensive', 'was', 'already', 'very', 'entrenched', 'by', 'then', '.']
srl_frames: ['O', 'O', 'O', 'O', 'O', 'B-ARG1', 'O', 'O', 'O', 'O', 'O', 'O', 'O']
verb_index: 4
words_indices: [13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25]
Labeled srl_frames: [0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0]
```

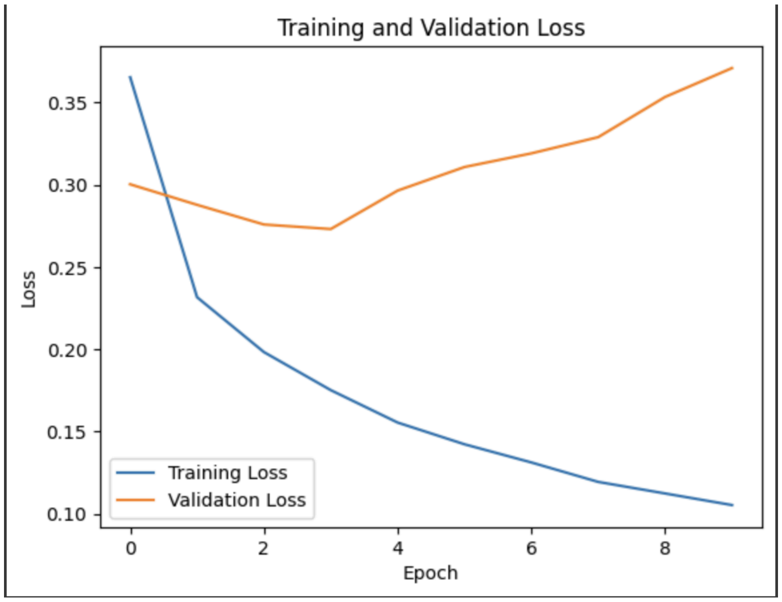
Vocab class implementation:

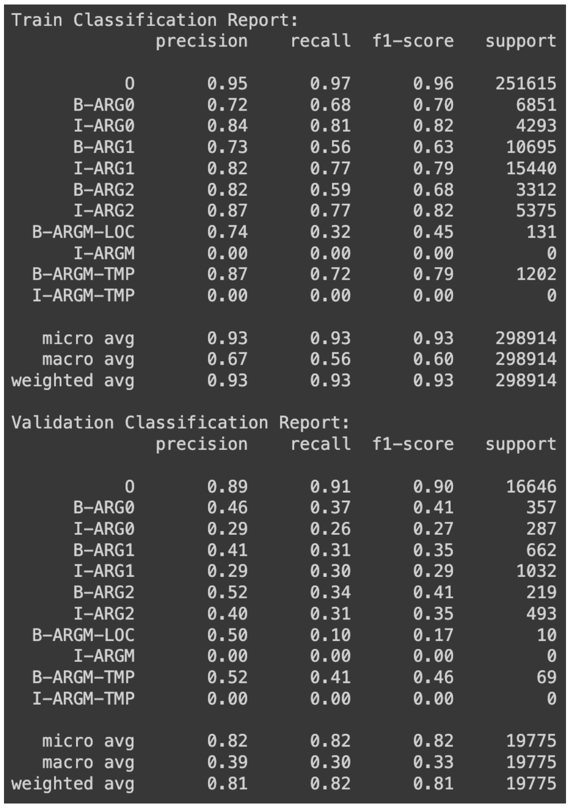
```python
class Vocab:

    def __init__(self, word2id=None):

        if word2id is None:

            self.word2id = {

                "<PAD>": 0,

                "<START>": 1,

                "<END>": 2,

                "<UNK>": 3

            }

        else:

            self.word2id = word2id

        self.id2word = {id: word for word, id in self.word2id.items()}


    def __getitem__(self, word):

        return self.word2id.get(word, self.word2id["<UNK>"])


    def __len__(self):
        return len(self.word2id)


    def add(self, word):

        if word not in self.word2id:

            index = len(self.word2id)

            self.word2id[word] = index

            self.id2word[index] = word

        return self.word2id[word]


    def word2indices(self, sents):

        return [[self[word] for word in sent] for sent in sents]


    def indices2words(self, word_ids):

        return [self.id2word[id_] for id_ in word_ids]


    @classmethod

    def to_input_tensor(cls, batch):

      sentences, verb_indices, labels = zip(*batch)

      sentences_padded = pad_sequence(sentences, batch_first=True,
padding_value=vocab['<PAD>'])

      labels_padded = pad_sequence(labels, batch_first=True,
padding_value=srl_data_handler.label_dict['O'])

      lengths = torch.tensor([len(sentence) for sentence in sentences])


      return sentences_padded, torch.tensor(verb_indices), labels_padded,
lengths
```

```python
    @classmethod
    def from_corpus(cls, corpus, size, freq_cutoff, remove_frac):
        word_freq = Counter(word for sent in corpus for word in sent)
        filtered_words = [word for word, freq in word_freq.items() if freq
>= freq_cutoff]

        filtered_words = sorted(filtered_words, key=lambda x:
-word_freq[x])
        if remove_frac > 0:
            num_remove = int(len(filtered_words) * remove_frac)
            filtered_words = filtered_words[:-num_remove]

        vocab = cls()
        for word in filtered_words[:size]:
            vocab.add(word)
        return vocab
```

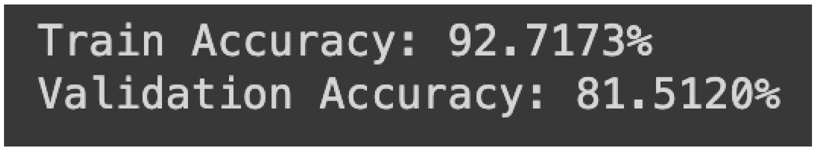
Answer of the second part:

Part 2-1 & 2-2:





```
Train Classification Report:
              precision    recall  f1-score   support

           O       0.95      0.97      0.96    251615
      B-ARG0       0.72      0.68      0.70      6851
      I-ARG0       0.84      0.81      0.82      4293
      B-ARG1       0.73      0.56      0.63     10695
      I-ARG1       0.82      0.77      0.79     15440
      B-ARG2       0.82      0.59      0.68      3312
      I-ARG2       0.87      0.77      0.82      5375
  B-ARGM-LOC       0.74      0.32      0.45       131
      I-ARGM       0.00      0.00      0.00         0
  B-ARGM-TMP       0.87      0.72      0.79      1202
  I-ARGM-TMP       0.00      0.00      0.00         0

   micro avg       0.93      0.93      0.93    298914
   macro avg       0.67      0.56      0.60    298914
weighted avg       0.93      0.93      0.93    298914

Validation Classification Report:
              precision    recall  f1-score   support

           O       0.89      0.91      0.90     16646
      B-ARG0       0.46      0.37      0.41       357
      I-ARG0       0.29      0.26      0.27       287
      B-ARG1       0.41      0.31      0.35       662
      I-ARG1       0.29      0.30      0.29      1032
      B-ARG2       0.52      0.34      0.41       219
      I-ARG2       0.40      0.31      0.35       493
  B-ARGM-LOC       0.50      0.10      0.17        10
      I-ARGM       0.00      0.00      0.00         0
  B-ARGM-TMP       0.52      0.41      0.46        69
  I-ARGM-TMP       0.00      0.00      0.00         0

   micro avg       0.82      0.82      0.82     19775
   macro avg       0.39      0.30      0.33     19775
weighted avg       0.81      0.82      0.81     19775
```



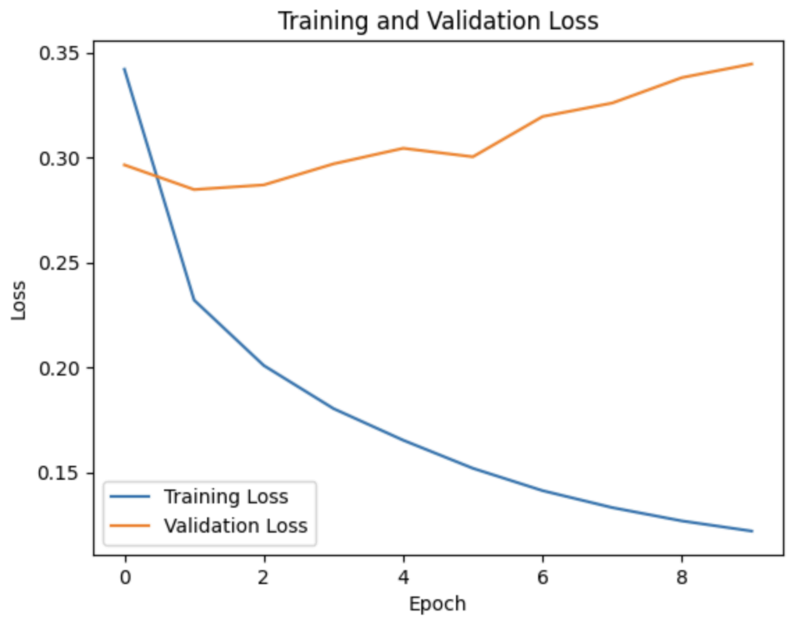
```
Train Accuracy: 92.7173%
Validation Accuracy: 81.5120%
```

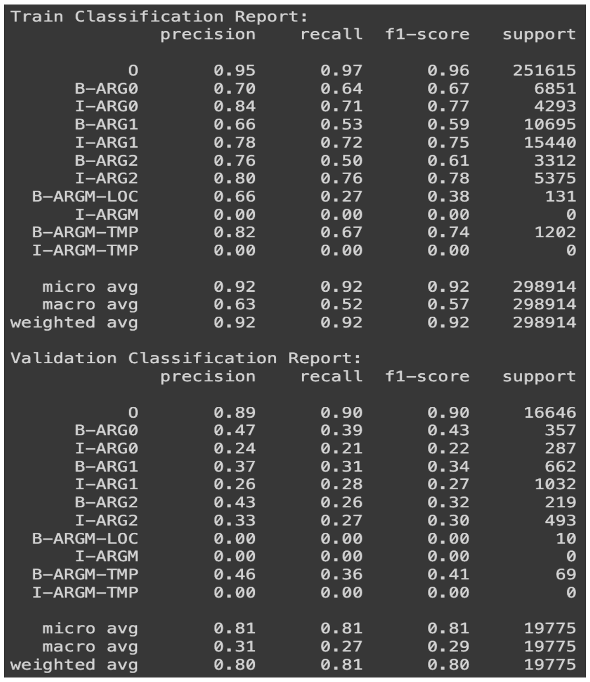
The plot shows the training and validation loss of model across epochs. The training loss decreases sharply and consistently as the number of epochs increases, indicating that the model is effectively learning from the training data. The validation loss initially decreases but then starts to increase, suggesting that the model may be beginning to overfit to the training data while its ability to generalize to unseen data is decreasing.

An F1 Score on the train set suggests that the model has a good balance between precision and recall during training. However, a significantly lower F1 Score on the validation set points to potential overfitting, as the model does not generalize well to unseen data. The accuracy of the model shows that with a high training accuracy of 92.7173% indicating that the model successfully captures the patterns in the training data. The validation accuracy is lower at 81.5120%, which reinforces the evidence of overfitting observed from the F1 Score.

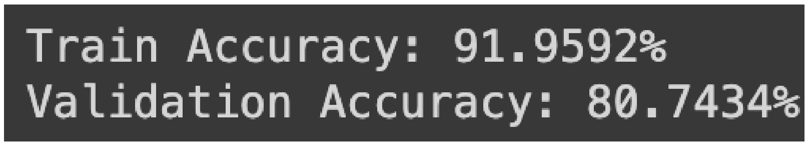
Answer of the third part:

Part 3-1:





```
Train Classification Report:
              precision    recall  f1-score   support

           O       0.95      0.97      0.96    251615
      B-ARG0       0.70      0.64      0.67      6851
      I-ARG0       0.84      0.71      0.77      4293
      B-ARG1       0.66      0.53      0.59     10695
      I-ARG1       0.78      0.72      0.75     15440
      B-ARG2       0.76      0.50      0.61      3312
      I-ARG2       0.80      0.76      0.78      5375
  B-ARGM-LOC       0.66      0.27      0.38       131
      I-ARGM       0.00      0.00      0.00         0
  B-ARGM-TMP       0.82      0.67      0.74      1202
  I-ARGM-TMP       0.00      0.00      0.00         0

   micro avg       0.92      0.92      0.92    298914
   macro avg       0.63      0.52      0.57    298914
weighted avg       0.92      0.92      0.92    298914

Validation Classification Report:
              precision    recall  f1-score   support

           O       0.89      0.90      0.90     16646
      B-ARG0       0.47      0.39      0.43       357
      I-ARG0       0.24      0.21      0.22       287
      B-ARG1       0.37      0.31      0.34       662
      I-ARG1       0.26      0.28      0.27      1032
      B-ARG2       0.43      0.26      0.32       219
      I-ARG2       0.33      0.27      0.30       493
  B-ARGM-LOC       0.00      0.00      0.00        10
      I-ARGM       0.00      0.00      0.00         0
  B-ARGM-TMP       0.46      0.36      0.41        69
  I-ARGM-TMP       0.00      0.00      0.00         0

   micro avg       0.81      0.81      0.81     19775
   macro avg       0.31      0.27      0.29     19775
weighted avg       0.80      0.81      0.80     19775
```



```
Train Accuracy: 91.9592%
Validation Accuracy: 80.7434%
```

The results are a GRU-based semantic role labeling model. The training loss steadily decreases, which is typically expected as the model learns and optimizes its weights. The validation loss, after initially dropping, begins to rise, which may indicate that the model is starting to overfit the training data.

The F1 Score on the train set is quite high, suggesting good predictive performance on the data used to fit the model. However, the much lower F1 Score on the validation set points towards a substantial performance gap when the model is applied to unseen data, highlighting issues with overfitting or insufficient model generalization also the model's accuracy is quite high on the training set at about 92.7173%, indicating that the model is accurate in classifying the correct semantic roles on the data it has seen during training. The validation accuracy is significantly lower at about 81.512%, which, similar to the F1 Score, suggests that the model is not performing as well when faced with new data, and the accuracy drop corroborates the possibility of overfitting.

Part 3-2: (Answer of the Questions)

1: What is the advantage of LSTM over RNN?

LSTM networks were developed to address deficiencies in traditional RNNs, such as the vanishing gradient problem and the difficulty in retaining information over long sequences. RNNs struggle to maintain memory for long periods, which affects their performance on tasks requiring long-term dependencies. LSTMs overcome this with their gate mechanisms which regulate information flow, allowing them to maintain memory for longer periods and learn dependencies across extensive sequences more effectively.

2: Explain the difference between LSTM and GRU.

Both LSTM and GRU tackle the limitations of traditional RNNs and incorporate gating mechanisms, but they differ in complexity and structure. GRU simplifies the LSTM architecture by using only two gates: the update gate and reset gate, compared to the three gates (input, forget, and output) used in LSTMs. This simplicity generally makes GRUs **faster** to train and can **perform better or comparably on tasks that do not require the sophisticated gating mechanisms** of LSTMs, particularly when dealing with less data or where computational efficiency is crucial.

3: Why do we need to concatenate the current hidden state with the hidden layers of all tokens in this model?

Concatenating the current hidden state with the hidden states of other tokens enhances model performance by providing a more nuanced understanding of context and dependencies between tokens. This method improves the model's decision making ability in accurately recognizing roles and extracting detailed syntactic and semantic information from sentences.

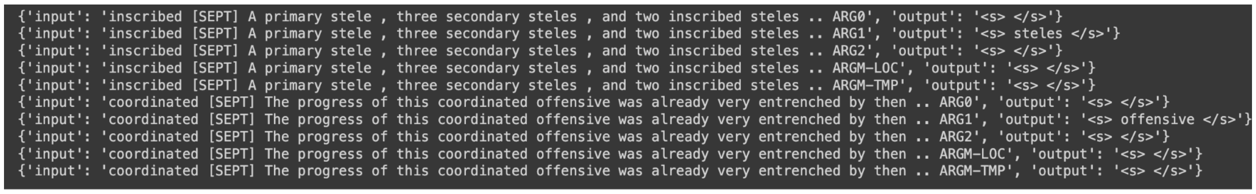
4: What solutions would you suggest if you encounter the problem of vanishing gradients in recurrent networks (without changing the model itself)?

Adjusting the learning rate: Using a smaller learning rate can prevent drastic weight changes, which sometimes exacerbate the vanishing gradient issue.

Gradient Clipping: This technique limits the size of the gradients to a defined threshold, thereby preventing the gradients from becoming too large or exploding, which helps maintain stability in the network's training process.

Answer of the fourth part:

Part 4-1:



```
{'input': 'inscribed [SEPT] A primary stele , three secondary steles , and two inscribed steles .. ARG0', 'output': '<s> </s>'}
{'input': 'inscribed [SEPT] A primary stele , three secondary steles , and two inscribed steles .. ARG1', 'output': '<s> steles </s>'}
{'input': 'inscribed [SEPT] A primary stele , three secondary steles , and two inscribed steles .. ARG2', 'output': '<s> </s>'}
{'input': 'inscribed [SEPT] A primary stele , three secondary steles , and two inscribed steles .. ARGM-LOC', 'output': '<s> </s>'}
{'input': 'inscribed [SEPT] A primary stele , three secondary steles , and two inscribed steles .. ARGM-TMP', 'output': '<s> </s>'}
{'input': 'coordinated [SEPT] The progress of this coordinated offensive was already very entrenched by then .. ARG0', 'output': '<s> </s>'}
{'input': 'coordinated [SEPT] The progress of this coordinated offensive was already very entrenched by then .. ARG1', 'output': '<s> offensive </s>'}
{'input': 'coordinated [SEPT] The progress of this coordinated offensive was already very entrenched by then .. ARG2', 'output': '<s> </s>'}
{'input': 'coordinated [SEPT] The progress of this coordinated offensive was already very entrenched by then .. ARGM-LOC', 'output': '<s> </s>'}
{'input': 'coordinated [SEPT] The progress of this coordinated offensive was already very entrenched by then .. ARGM-TMP', 'output': '<s> </s>'}
```

Notice: model implementation is in part4.py file but it does not run properly and I could not find results for this part.

Part 4-4:

What are the limitations of the method of converting the SRL problem into a QA problem using the encoder-decoder model?

The model's performance heavily depends on the quantity and quality of the training data. Inadequate or biased data can lead to inaccurate or skewed predictions and also encoder-decoder architectures, particularly those using attention mechanisms and LSTMs, can be computationally expensive to train and require significant GPU resources for timely training. SRL inherently involves identifying specific roles linked to predicates within sentences. By transforming it into a QA format, some nuances and granular details typical in SRL tagging might be oversimplified or lost in the translation to Q&A pairs. SRL requires a deep understanding of context, not just surrounding words but also the broader sentence or paragraph meaning. While QA systems can handle context to an extent, the focus on querying specific information can sometimes lead to overlooking broader contextual cues critical for correct role assignment.

Why do we use <s/> and <s> tokens at the beginning and end of the output when training the model?

The use of <s> and </s> tokens is crucial for managing the lifecycle of sequence generation by indicating the start and end of a text, thus helping the model understand and produce structured and appropriately bounded output. By including these tokens in the set used to train embeddings such as GloVe, all tokens used in training and inference have a consistent representation. This is key to maintaining the integrity of the input data throughout the model's architecture. Each token, whether a common word, a rare word, or a special marker like <s> and </s>, has a corresponding vector in the embedding space.

Answer of the fifth part:

Due to the lack of the results of the fourth part, it is not possible to compare.