Arie Cattan 336319314
Moche Uzan F957022

Part 3

First model using embedding vector

We use Pytorch to build a transducer model with the following parameters:

- One embedding layer of 50
- A first bidirectional LSTM with hidden layer of 50
- A second bidirectional LSTM with hidden layer of 50
- A linear layer to the tagset size (47 for POS data ,6 for NER data)
- A softmax over the last layer

We choose from the output the best class as prediction.

We train it with an Adam optimizer through 5 epoch and using mini-batch of 20.

We use the cross entropy loss and a learning rate of 0.01 with a decay of 0.5 after each epoch

Second model using a character-level LSTM

First we make a character level model following this parameters:

- One embedding layer of 10
- A dropout layer (probability= 0.5)
- An LSTM with hidden layer of 50 (we return the sum of the char to have one embedding by word)

We passed over each word and replace each character by its id and we pad with 0 to make a word length constant.

And then we build a transducer model that use the first model with the following parameters:

- The character level model
- A first bidirectional LSTM with hidden layer of 50 and a dropout (probability= 0.5)
- A second bidirectional LSTM with hidden layer of 50 and a dropout (probability= 0.5)
- A linear layer to the tagset size (47 for POS data ,6 for NER data)
- A softmax over the last layer

We choose from the output the best class as prediction.

We train it with an Adam optimizer through 5 epoch and using mini-batch of 20.

We use the cross entropy loss and a learning rate of 0.01 with a decay of 0.5 after each epoch

Arie Cattan 336319314
Moche Uzan F957022

Third model using embedding and subword representation

- An embedding layer for the word and its subwords of 50. For the subwords, we took the 3 first letters as prefix and 3 last as suffix. We then added the embedding of the words and the subwords.
- A first bidirectional LSTM with hidden layer of 50
- A second bidirectional LSTM with hidden layer of 50
- A linear layer to the tagset size (47 for POS data ,6 for NER data)
- A softmax over the last layer

We choose from the output the best class as prediction.

We train it with an Adam optimizer through 5 epoch and using mini-batch of 100.

We use the cross entropy loss and a learning rate of 0.01 with a decay of 0.5 after each epoch, and we found it's the best model.

Fourth model using a character-level LSTM

First we forward the sentence over the first and the second model.
And then we build a transducer model that use their output and forward them by:

- Concatenate the two output
- Dropout layer (probability = 0.5)
- A linear layer to the tagset size (47 for POS data ,6 for NER data)

We choose from the output the best class as prediction.

We train it with an Adam optimizer through 5 epoch and using mini-batch of 20.

We use the cross entropy loss and a learning rate of 0.01 with a decay of 0.5 after each epoch

Arie Cattan 336319314
Moche Uzan F957022

Result

We plot the evolution of the accuracy over the iterations.

We can observe that the accuracy has some drop, because the PyTorch LSTM is fed by PackedSequence which require the sorting of the sentence so at each begin of epoch the sentences are longer.