COMP6714 18s2 Project

Stage 1: Implement a hyponymy classification model using BiLSTM

Objective ¶

In this project, you need to build a system that can extract hyponym and hypernym (https://en.wikipedia.org/wiki/Hyponymy_and_hypernymy) from a sentence.

For example, in sentence *Stephen Hawking is a physicist .*, phrase *Stephen Hawking* is the **hyponym** of *physicist*, and *physicist* is the **hypernym** of *Stephen Hawking*.

We formulate the problem as a sequence tagging task where

- Input is a sentence formed as a sequence of words with length l.
- output is a tag sequence with length l.

We use IOB2 (https://en.wikipedia.org/wiki/Inside%E2%80%93outside%E2%80%93beginning_(tagging)) scheme to represent the results (i.e., encode the tags), as both hypernyms and hyponyms can be phrases.

Thus, in the above example,

- `[Stephen, Hawking, is, a, physicist, .]` is the input word list
- `[B-TAR, I-TAR, O, O, B-HYP, O]` is the corresponding output tag sequence, where `TAR` corresponds to hyponyms and `HYP` corresponds to hypernyms.

You need to build and train a neural network model for this task.

Data Preprocessing

Description of given files

- `tags.txt` lists all the tags.
- `train.txt` , `dev.txt` , `test.txt` refers to training set, development set, and test set, individually. Each of them is in the CoNLL format, with each line having two columns for word and tag. Sentences are separated by blank lines.
- `word_embeddings.txt` is the pretrained word embedding file, with word in the first column and embeddings (a vector with 50-dimensions) in the rest columns.

Generate batches

- In order to speed up the training process, instead of pass the training sentences one by one to the model, we pass a batch of sentences at each iteration.

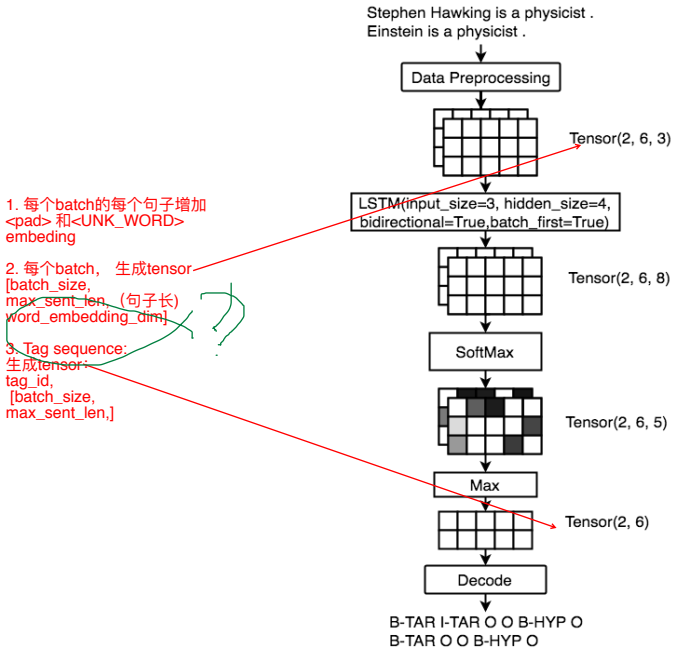
Generate tensors

- The first step is to convert each sentence into a list of word embeddings. In order to do that, we need two additional *words*, they are

    - `<PAD>` is used to pad words to the end of sentences, such that all the sentence in the same batch have the same length. For example, if there are 5 sentences in a batch, with length `[3, 4, 4, 5, 5]`, then we need to pad two `<PAD>` for the first sentence and one `<PAD>` for the second and third sentences.
    - `<UNK_WORD>` is used for those words that can not be found in word embedding.
    - we offered the embedding of `<UNK_WORD>` in the embedding file. And `<PAD>` can be randomly initialled. It will not contribute to the loss function or the prediction, thus will not be updated anyway.
- Then for each batch, we can generate the corresponding **tensor** with size `[batch_size, max_sent_len, word embedding dim]`. Where `max_sent_len` is the maximum length among sentences in the batch.
- For tag sequence, We generate a **tensor** with size `[batch_size, max_sent_len]`, in which each element represents a tag_id.

A Baseline BiLSTM Model

- In this project, you are required to use Bidirectional LSTM (BiLSTM) and SoftMax in your model.
- The input tensor will go through a *BiLSTM layer*, followed by a *softmax* function to determine the output tags.
- An example of the model is shown as follows:



.

Training

In the training process, in each epoch, you should feed the training data to the model batch by batch, and update the paramters accordingly.

You should use ADAM as the optimizer for your model.

If your implementation is correct, then your loss should converge within 80 epochs. Therefore the maximum number of epochs is set to 80.

You can either take the model from the last epoch or the one with highest F1 score on the development set as the final model.

Helpful Resources

- PyTorch Documents (https://pytorch.org/docs/stable/index.html)
- Sequence Modelling Tutorial (https://pytorch.org/tutorials/beginner/nlp/sequence_models_tutorial.html)

Stage 2 for the project

In the next stage of the project, you will be given a baseline model based on the above description. You will need to firstly read and understand the implementation, and then finish the following 3 request:

- implement a special LSTM cell and replace it with the default one
- implement character embedding and plug it into the model
- implement the evaluation function (i.e., compute F1 score) to help the training process with model selection.

You are also expected to write a short report with the implementation details.

Detailed specification of stage 2 will be posted later.