Sequence to Sequence Parsing

CODE STRUCTURE

The entire project is submitted in the folder *project4*, which has two subdirectories:

- *nmt* – the sequence2sequence code, taken from https://github.com/tensorflow/nmt#introduction
  - *nmt* – folder containing the actual code
  - *tmp* – folder containing the prepared data (prepared using code in *data* directory and manually placed in *tmp/parse_data*):
    - *EXPERIMENTS* – stores results of previous runs
    - *parse_data*: stores the required data with corresponding extensions (files were prepared by *prepare_input.py* code and transferred here):
      - *dev.en*
      - *dev.parse*
      - *test.en*
      - *test.parse*
      - *train.en* (and *train_150000_sentences.en*)
      - *train.parse* (and *train_150000_sentences.parse*)
      - *vocab.en*
      - *vocab.parse*
- *data* – this contains packages and scripts for processing and evaluating data (and data itself):
  - *train*, *dev*, *test* – folders with gold data
  - *dev_sents* – single text file with all dev sentences (used for EVALB)
  - *test_sents* – single text file with all test sentences (used for EVALB)
  - *EVALB* – bracket scoring code, taken from https://nlp.cs.nyu.edu/evalb/
  - *obsolete* – this contains my unfinished attempt at setting up a seq2seq model
  - *prepare_input.py* – code for converting parsed data into linearized format presented in class slides (actual words removed), which is then used in seq2seq model:
    - *e.g.* (S (NP (N John)) (VP (VBD sneezed))) → (S (NP N)NP (VP VBD)VP)S
  - *post_process.py* – code for reconverting the linearized output of the seq2seq model into standard parse trees (including inserting the vocabulary from corresponding sentence)
  - *seq2seq_output* – folder storing output parses of test data from the nmt model, transferred manually from the *parse_model* folder that is created during the run of *nmt*
  - ***postprocessed_output*** – folder with the outputs from *seq2seq_output* converted to the standard parsed format **– PLEASE RUN EVALB ON THE FILES IN THIS FOLDER!!!**

HOW TO RUN THE CODE

Running the *nmt* sequence to sequence program

Navigate inside the outer *nmt* directory, and run the following command (hyperparameters can be modified as desired):

```
python -m nmt.nmt
    --src=en --tgt=parse
    --vocab_prefix=./tmp/parse_data/vocab
    --train_prefix=./tmp/parse_data/train
    --dev_prefix=./tmp/parse_data/dev
    --test_prefix=./tmp/parse_data/test
    --out_dir=./tmp/parse_model
    --num_train_steps=12000
    --steps_per_stats=100
    --num_layers=2
    --num_units=128
    --dropout=0.2
    --metrics=bleu
```

In the process a folder named *parse_model* will be created in *tmp* (as specified in settings above), inside which the predicted *output_dev* and *output_test* files will appear after training. After the program terminates, transfer these files to *seq2seq_output* inside the *data* directory (I have only done this for *output_test*, which I evaluated).

Running *post_process.py*

When running *post_process.py*, you must provide the filepath to the output of the *nmt*, located in *seq2seq_output*, as well as the filepath to a file in *postprocessed_output* where an empty file of the corresponding waits to be written to. [Since I ran the program from PyCharm, I changed the filepaths right in the main method instead of providing *sys.argv* (this can be easily modified).] The resulting post-processed file will be used by EVALB.

Some notes on *post_process.py*: this code reconverts a sequence such as *(S (NP N)NP (VP VBD)VP)S* back to a normal tree for the corresponding sentence (i.e. *John sneezed*). While it is always possible to remove the labels from the closing brackets, the output doesn't always give the correct number of parts of speech (such as *VBD*) to match the number of words in the corresponding sentence. Therefore my code simply iterates simultaneously over the parts of speech in the output parse and the corresponding words in the original sentence, and inserts a word where possible (i.e. *VBD → (VBD sneezed)*). If there are more words than parts of speech, the extra words don't get inserted. If there are more parts of speech than words in the sentence, my code inserts a dummy '-' character as the word for the POS.

Evaluating Accuracy Using EVALB

Navigate inside EVALB and run:

```
./evalb -e 50000 ../test_sents ../postprocessed_output/postprocessed_output_test_default
```

where we provide the [relative] path first to the file with all the test sentences (*test_sents* – the first argument) and then the post-processed output of the seq2seq model (the second argument). Please provide a high maximum error threshold (e.g. 50000, as in the command above) to avoid EVALB stalling after a high number of invalid sentences.
