5. Practical exercise

Introduction to Deep Learning

R. Lasowski

Winter term 2018/19

Practicals to Neural Machine Translation

Aufgabe P 5. *Neural Machine Translation, due 11 January in class*

In this practical you will implement an encoder-decoder neural network for neural machine translation. This practical is based on the tutorial from https://github.com/tensorflow/nmt. The data that you will train on are text files that include conversations (En/Deu) or just words (En/Deu) of translated sign language videos. Each group should add five test data sets. You should use words that appear in the dictionary.

(a) Build your german and english lower-case vocabulary as a dictionary. In Python you can use the *dict* object. With this data structure you can get the word via the index. Then you should generate an inverted dictionary: get the index via a word. Add the special tags start and end of sentence: '¡s¿' and '¡/s¿'.

- Process first the iso*.txt files. For example iso0077.txt has following lines:

  annot_eng COST
  annot_deu KOSTEN

  Add 'kosten' to your german dictionary, 'cost' to your english dictionary and ignore 'annot_deu' and 'annot_eng'. Another example:
  annot_eng MY|PREFERRED
  annot_deu MEIN|LIEBER
  Add 'my', 'preferred' as two separate words to your english dictionary. Proceed the same with the german words.
- Process the con*.txt files. Example:

  annot_eng EXCUSE-ME YOU SEE MY IDENTITY-CARD STEAL WHO?
  annot_deu ENTSCHULDIGUNG DU SEHEN MEIN AUSWEIS STEHLEN WER?
  transl_eng Excuse me, did you see the person who has stolen my identity card?
  transl_deu Entschuldigung, hast du gesehen, wer meinen Ausweis gestohlen hat?

  For the con*.txt file you ignore the first two lines and then start adding: 'excuse', 'me', 'did'... to your english dictionary. Proceed the same way with the german words.

(b) Define your inputs and outputs

- encoder_inputs [batch_size, max_encoder_time]: source input words.
- decoder_inputs [batch_size, max_encoder_time]: target input words.
- decoder_outputs [batch_size, max_encoder_time]: target output words, these are decoder_inputs shifted to the left by one time step with an end-of-sentence tag appended on the right.

as placeholder: $encoder\_inputs = tf.placeholder(tf.int32, shape = [batch\_size\_training, None], na$ $enc\_in')$. You can use a $batch\_size\_training$ of one.

(c) Example:

- $[[400, 50, 18, 9, src\_eos\_id]]$ source input words, where $src\_eos\_id = dict\_deu['< /s >']$
- $[[tgt\_sos\_id, 7, 300, 15, 9]]$ target input words, where $tgt\_sos\_id = dict\_en['< s >']$
- $[[7, 300, 15, 9, tgt\_eos\_id]]$ target output words, $tgt\_eos\_id = dict\_en['< /s >']$

(d) Follow the tutorial to build your model. Start a session for training. Feed your placeholders with the sentences encoded as integers via the dictionaries.

(e) Once you trained your system start a session for evaluating your test sentences. Print your translations and inspect them. We don't use here a metric for evaluation.