Sequence-to-Sequence Learning

Chapter 5 – (Introduction to Deep Learning)
Introduction

• Sequence to sequence learning is a deep learning technique to map a sequence of symbols to the other.
  • Used specifically when mapping symbols individually is not possible.
  • Machine Translation is the typical application of this algorithm.

• Word by word translation does not help in most of the cases.

| this being the day on which parliament was convoked by proclamation of his excellency ... |
| parlement ayant été convoqué pour aujourd ’ hui , par proclamation de son excellence ... |
Sequence-to-Sequence Paradigm

• The model consists of two RNNs.
• Uses GRU – Gated Recurrent Unit which passes a single memory line between time units.
• Total two passes:
  • Encoding: First phase of the seq2seq process, set of symbols (in source language) are passed through GRU. The goal of this pass is to produce a sentence embeddings which summarizes the sentence.
  • Decoding: Second phase of the seq2seq process, passing through symbols in target language. The goal of this pass is to predict the word/symbol after each word/symbol is input.
Sequence-to-Sequence Paradigm
Sequence-to-sequence Paradigm

• The image is shown as back propagation through time.
  • So all the RNN units at the bottom row are actually the same recurrent unit but at successive time.
  • Same goes for the units in the top as well.

• The book assumes a few simplifications:
  • Every sentences starts and ends with STOP word.
  • Machine translation complexity is ignored to predict the next word given the previous word.
  • All sentences are limited to the length of 13 words. Sentences shorter are padded with extra STOP word(s).
Sequence-to-sequence Program

```
1  with tf.variable_scope("enc"):  
10 with tf.variable_scope("dec"):  
```

- Variable scope is used here to differentiate the encoding and decoding parts of the program.
  - It helps with TF program design as well because TF uses same name variable to insert into TF graph while working with multiple dynamic_rnn calls.
  - Variable scope stops multiple dynamic_rnn calls step on each other and avoid error.
  - Lines in above diagram, defines scope for encoding and decoding respectively.
Sequence-to-sequence Program

2    F = tf.Variable(tf.random_normal((vfSz, embedSz), stddev=.1))
3    embs = tf.nn.embedding_lookup(F, encIn)
4    embs = tf.nn.dropout(embs, keepPrb)
5    cell = tf.contrib.rnn.GRUCell(rnnSz)
6    initState = cell.zero_state(bSz, tf.float32)
7    encOut, encState = tf.nn.dynamic_rnn(cell, embs,
8                                        initial_state=initState)

- First, the French word embeddings are created of by passing the French word indices in the shape of batch size by window size.
- The lookup function will return the 3D tensor of batch size by window size by embedding size.
Sequence-to-sequence Program – cont’d

• Dropout with probability of keeping the connection is applied to the output of the lookup.
• Then RNN cell is created through GRU variant of LSTM.
• This cell is used to create output and the next state through dynamic RNN.

```python
11   E = tf.Variable(tf.random_normal((veSz,embedSz), stddev=.1))
12   embs = tf.nn.embedding_lookup(E, decIn)
13   embs = tf.nn.dropout(embs, keepPrb)
14   cell = tf.contrib.rnn.GRUCell(rnnSz)
15   decOut,_ = tf.nn.dynamic_rnn(cell, embs, initial_state=encState)
```
Sequence-to-sequence Program – cont’d

• In the decoder variable scope, the output of the encoding RNN is used by the `dynamic_rnn` of the decoder.

• The output of decoder also feeds into loss computation which can be done using below code.

```python
W = tf.Variable(tf.random_normal([rnnSz, veSz], stddev=.1))
b = tf.Variable(tf.random_normal([veSz, stddev=.1]))
logits = tf.tensordot(decOut, W, axes=[[2], [0]]) + b
loss = tf.contrib.seq2seq.sequence_loss(logits, ans, tf.ones([bSz, wSz]))
```

• `Seq2seq.sequence_loss` is a specialized version of cross-entropy loss.
Sequence-to-sequence sentence summary

• The idea of seq2seq translation is to create a summary of a sentence in source language by passing it through GRU.

• There are multiple ways to create sentence summaries. Instead of passing just the encoder state output, sum of all encoder states can be passed to decoder.

• Another possibility is to pass the average value of all encoder states instead of passing the sum.

• According to the book, passing the sum seems to more informative compared one final vector.
Attention in Sequence-to-sequence

• Concept of attention in seq2seq comes from the fact that a patch of target word translations depend more on some part of the source sentence than the other.

• Encoding phase output is fed into all decoder states indicating equal importance to all states.
  • For attention models, some encoding states’ output are mixed together in different proportion before feeding them into decoding scope.
  • This is called ‘position-only attention’.

• For attention model scheme, attention paid by the word of source language at position i to the target language word at position j depends only on i and j. Attention is higher for close i and j.
Multilength Sequence-to-sequence

• One of the simplification considered for seq2seq is to have sentence limited to 13 words.
  • In reality, this is very small limit but increasing the limit might impact sentences with less number of words.
• In the seq2seq program, dynamic_rnn call takes in embds which is of dimension, batch size by window size by embed size.

```python
cell = tf.contrib.rnn.GRUCell(rnnSz)
encOutSmall, encStateS = tf.nn.dynamic_rnn(cell, smallerEmbs, ...)
encOutLarge, encStateL= tf.nn.dynamic_rnn(cell, largerEmbs, ...)
```

• Because there is a single GRU cell which is used for smaller and larger window size, they learn and share same knowledge for source and target language.