Sequence to Sequence Modelling

Patrick Kahardipraja

September 2019

Mapping of a sequence to another sequence is an important paradigm because of the vast amount of problems that can be formulated in this manner. For instance, in automatic speech recognition (ASR), chunks of speech signals can be mapped to sequence of phonemes while in machine translation, a sequence of words in one language can be mapped to another language. Interestingly, many other tasks such as text summarization, question answering, and image caption generation can be phrased as a sequence to sequence problem. In this paper, I will attempt to distill how sequence to sequence learning works and the motivation behind it, with a particular focus on machine translation.

Introduction

Prior to neural machine translation (NMT), phrase-based stastistical machine translation (SMT) systems are widely used as it offers reliable performance. Despite its success, most of them are extremely complex and require a huge amount of effort, as it is often tailored to a specific language pair and do not generalize well to another languages. Furthermore, a lot of feature engineering are required in order to capture a specific language phenomena, which prompt researchers to explore another approach.

The resurgence of deep neural networks (DNNs) in early 2010s, thanks to faster, parallel computation using GPUs and availability of large and high-quality datasets, bring a new wave of enthusiasm in deep models. With the capability to learn features automatically with multiple, hierarchical representation, DNNs achieve excellent performance on difficult tasks in computer vision [AlexNet] and speech recognition []. Albeit powerful, DNNs has its own limitation, as it requires input and output vectors with a fixed dimension and thus not suitable for sequence to sequence problem whose lengths are unknown beforehand. In addition, DNNs also do not generalize well across temporal patterns, because each neuron has its own specific connection and as a result, a single pattern may look totally different at different timesteps.

The natural remedy for this problem is to look onto recurrent neural networks (RNNs), as it allows operations over sequences of vectors. However, mapping using RNNs typically have one-to-one correspondence between the input vectors and the output vectors. It also has another problem, as the input and output sequences can have different lengths and non-monotonic alignments. Standard RNN architecture is also not reliable for learning long-range dependencies due to the vanishing gradient problem. This issue is addressed by Sutskever et al. [Seq2Seq], where they introduce a novel and straightforward method to solve general sequence to sequence mapping using Long-Short Term Memory (LSTM) architecture. With the success of sequence to sequence learning in machine translation tasks, research in neural machine translation continue to thrive, eventually resulting in many significant improvements such as attention mechanism [Bahdanau] and subword units to deal with rare words [WordPiece]. But, before delving in too deep, I will give some brief insight into the mechanism behind RNN in the next section.

Recurrent Neural Networks

Recurrent neural networks [Rumelhart] are type of neural network that is able to process arbitrary sequential input via combination of its internal state and input vector. At every timestep t, the hidden state vector h_t is overwritten as a function of the hidden state at the previous timestep h_{t-1} and the current input vector x_t. The input vector x_t itself could be a representation of t-th word in a sentence, which is usually obtained using pre-trained word embeddings [GloVe, Word2Vec, ElMo]. The hidden state of RNNs can be perceived as a memory with a fixed dimensionality that can be tuned, containing distributed representation of the processed input sequence up to time t.

In a RNN, the forward step function consists of an affine transformation followed by non-linear activation function. The hidden state then can be used to make predictions:

$$h_t = a(W_x x_t + W_h h_{t-1} + b_h)$$
$$y_t = g(W_y h_t + b_y)$$

where in a typical application, a is the hyperbolic tangent function and g is the softmax function.

Insert pic of RNN and LSTM!!

Seq2Seq Model

The first work [Cho et al.]

Recent Advances

Conclusion