Sequence to Sequence Learning

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 a 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-style essay, 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 statistical 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 they are often tailored to a specific language pair and do not generalize well to other languages. In a phrase-based

SMT system, a lot of feature engineering is required in order to capture a specific language phenomenon, which prompt researchers to explore another approach. Furthermore, phrase-based systems still experience difficulty in capturing long-term dependencies.

The resurgence of deep neural networks (DNNs) in the 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 (Krizhevsky et al., 2012) and speech recognition (Hinton et al., 2012). Albeit powerful, DNNs have its own limitation, as they require 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 they allow operations over sequences of vectors. However, mapping using RNNs typically has a one-to-one correspondence between the input vector and the output vector. It also has another problem, as the input and output sequences can have different lengths and non-monotonic alignment. 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., 2014), 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 et al., 2015) and sub-

word units to deal with rare words (Sennrich et al., 2016b).

Recurrent Neural Networks

Recurrent neural network (Rumelhart et al., 1986) is a 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 (Mikolov et al., 2013; Pennington et al., 2014; Peters et al., 2018). 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 a non-linear activation function. The hidden state then can be used to make predictions:

$$h_t = a(W^{(x)}x_t + W^{(h)}h_{t-1} + b_h) \tag{1}$$

$$y_t = g(W^{(y)}h_t + b_y) \tag{2}$$

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

Although proven to be effective, RNN still has its own shortcoming. The main problem with RNN is that during training, the magnitude of gradient can get weaker or stronger exponentially when backpropagating the error through time, especially with long sequences (Hochreiter, 1998; Bengio et al., 1994). This phenomenon is called vanishing or exploding gradient problem,

which causes RNN model to experience difficulty when handling "long-term dependencies" that occur in a sequence.

Long Short-Term Memory (LSTM) architecture (Hochreiter and Schmidhuber, 1997) addresses the problem of "long-term dependencies" by integrating a memory cell that is capable to memorize state that span over long sequences of time. The memory cell is controlled by gates, which have the ability to regulate how much information is added or removed in the memory cell. This means that while in a RNN a completely new hidden state is computed at every new timestep, in LSTM the hidden state is not completely overwritten, and updated according to the memory cell. The architecture of both RNN and LSTM are depicted in Figure 1.

A LSTM unit consists of 3 gates (input gate i_t, forget gate f_t, output gate o_t), memory cell C_t and hidden state h_t. In a high-level sense, the input gate decides how much and which value will be updated, the forget gate controls the amount of information to be forgotten in the previous memory cell, and the output gate decides the hidden state by filtering the internal memory cell for each timestep. Each gate produces vector, which values are between 0 (completely closed) and 1 (completely open) using the sigmoid activation function.

The formula of LSTM is described with the following equations:

$$i_t = \sigma(W^{(i)}x_t + U^{(i)}h_{t-1} + b_i) \tag{3}$$

$$f_t = \sigma(W^{(f)}x_t + U^{(f)}h_{t-1} + b_f) \tag{4}$$

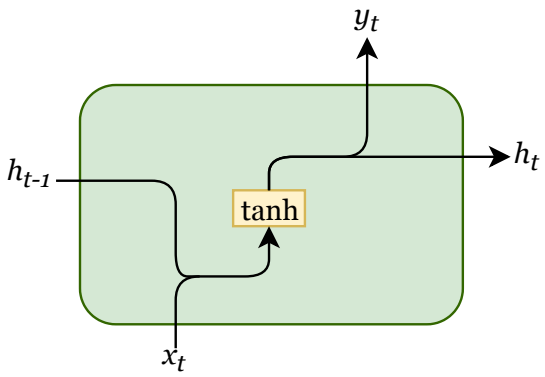
$$o_t = \sigma(W^{(o)}x_t + U^{(o)}h_{t-1} + b_o) \tag{5}$$

$$\tilde{C}_t = \tanh(W^{(c)}x_t + U^{(c)}h_{t-1} + b_c) \tag{6}$$

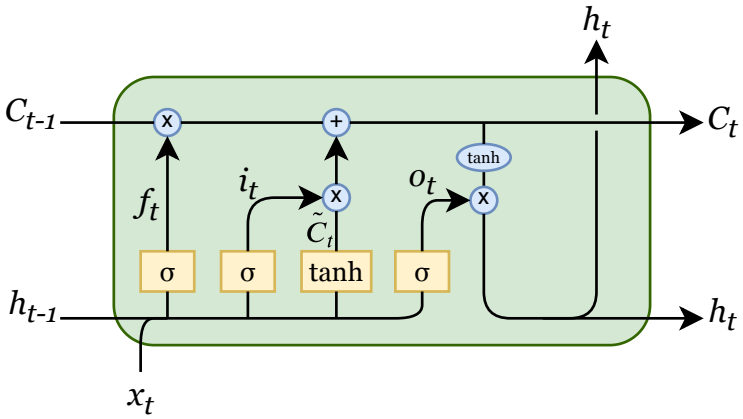
$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \tag{7}$$

$$h_t = o_t \odot \tanh(C_t) \tag{8}$$

where x_t is the input vector for timestep t, σ is the sigmoid activation function and $\odot$ denotes the Hadamard product of matrices.



(a) RNN unit



(b) LSTM unit

Figure 1: Architecture of RNN (a) and LSTM (b)

Variants of LSTM

Besides the standard LSTM architecture in the literature that is introduced above, there also exist several popular variants which are commonly used. One of the variations which is introduced by (Gers and Schmidhuber,

2000) use "peephole connections". These connections allow the gates to look into the memory cell state in order to learn precise and stable timings.

Other notable LSTM variant is the Gated Recurrent Unit (GRU), introduced by (Cho et al., 2014). Unlike LSTM, GRU is less complex and requires less computation. GRU only has 2 gates, the update gate that decides how much information will be transferred from previous and candidate hidden state to the current one and reset gate that controls to what extent the previous hidden state will affect the candidate hidden state. In this manner, the update gate can be thought of as a combination of input and forget gates of LSTM unit. This architecture also merges the internal memory cell and the hidden state of LSTM into a single hidden state.

Related Works

The first model that is able to map a sentence into a vector and then to its translation is introduced by (Kalchbrenner and Blunsom, 2013). The model, which they called Recurrent Continuous Translation Models (RCTM), is composed of 2 separate parts: convolutional neural network (CNN) for modeling the source sentence as the encoder and recurrent neural network for translation generation as a language modeling task, conditioned on the source sentence as the decoder. With this approach, the encoder can capture all the information contained in the source words and create a representation for the source sentences. The representation for the source sentences also restraints the generation of the target words in the language modeling phase.

The Recurrent Continuous Translation Models estimate the probability distribution over the sentence in the target language given a sentence in the source language. Suppose that there exists a target sentence $f = f_1, f_2, ..., f_m$, which is a translation of source sentence $e = e_1, e_2, ..., e_n$. Then

$p(f|e)$ can be obtained with the formula:

$$p(f|e) = \prod_{i=1}^{m} p(f_i|f_{1:i-1}, e) \tag{9}$$

As can be seen in the formulation above, the model estimates $p(f|e)$ by calculating the conditional probability $p(f_i|f_{1:i-1}, e)$ for every translated word occurring at position i, given the preceding generated words $f_{1:i-1}$ in the target sentence and the source sentence e. Conditioning the translation model to the preceding target words also ensures that it incorporates the target language model (Kalchbrenner and Blunsom, 2013).

In RCTM, prediction of the target sentence uses a language model based on a recurrent neural network (Mikolov et al., 2010). The recurrent language model (RLM) predicts the i-th word of the target sentence depending on all the previously generated words $f_{1:i-1}$, making no Markov assumption about the words dependencies in the target sentence. However, using the standard RNN architecture makes the prediction to be strongly affected by words close to f_i and weakly influenced by long-range dependencies that occur in the target language due to the nature of RNN.

The RLM models probability of a sequence of words f that occur in a language, denoted by $p(f)$. The equation for $p(f)$ is almost identical to Eq. 9 :

$$p(f) = \prod_{i=1}^{m} p(f_i|f_{1:i-1}) \tag{10}$$

It also contains a vocabulary V for words f_i of the language and 3 transformation matrices for input vocabulary $\mathbf{I}$, recurrent transformation $\mathbf{R}$ and output vocabulary transformation $\mathbf{O}$. Each word $f_k \in V$ is distinguished by

one-hot vector $v(f_k)$. The computation then proceeds as follows:

$$h_1 = g(\mathbf{I} \cdot v(f_1) + b_h) \tag{11}$$

$$h_{i+1} = g(\mathbf{R} \cdot h_i + \mathbf{I} \cdot v(f_{i+1}) + b_h) \tag{12}$$

$$o_{i+1} = \mathbf{O} \cdot h_i + b_o \tag{13}$$

and the probability distribution is obtained using the softmax function,

$$p(f_i = v | f_{1:i-1}) = \frac{\exp(o_{i,v})}{\sum_{v=1}^{V} \exp(o_{i,v})} \tag{14}$$

where g is a nonlinearity and $\mathbf{I} \cdot v(f_i)$ is a continuous representation of word f_i.

There are 2 types of conditioning architecture in RCTM with CNN, using convolutional sentence model (CSM) and convolutional n-gram model (CGM). The CSM creates sentence representation in a bottom-up manner, using n-grams representation in the sentence itself. The hierarchical structure that is created by the model acts quite similar like a parse tree in an implicit way. Using this type of structure, the model is able to capture the small, local representations in the lower layers of the model and more globally in the upper layers of the model as it spans more n-grams that comprise the sentence representations. This model also offers several advantages as it does not rely on a parse tree (Grefenstette et al., 2011; Socher et al., 2012). As there exist many languages for which highly accurate and reliable parsers are not available, this model can still be robustly applied when such case happens. Furthermore, the distribution of translation probability is learned by the model and does not depend on the chosen parse tree.

Using the continuous representations of words in the sentence, CSM models the representation of the sentence by applying sequences of one-dimensional convolution operations. The kernel of the convolutional layers

is able to learn pattern within n-grams that conveys syntactic, semantic or structural information relevant for constructing the sentence representation. After several convolution operations, the sentence vector representation $\mathbf{e}$ is created at the topmost layer of the network for the source sentence e. This vector representation is then used in the RLM, after applying learned sentence transformation $\mathbf{S}$. However, this model has a bias as the RLM tend to predict target sentences with shorter length. The sentence vector representation $\mathbf{e}$ also constraints the target words, which is counterintuitive as it often occurs that the target translation has a strong dependency on some parts of the source sentence and less on the other parts. In order to address these aspects, the convolutional n-gram model is also proposed as another conditioning architecture.

The CGM is a truncated version of the CSM, where the n-grams representation is extracted from a specific CSM layer for a chosen value of n. Using the n-grams representation of the source sentence e, the CGM can also be inverted to obtain representation for the target sentence f with a de-convolutional operation, where the length of the target sentence m is estimated using Poisson distribution. This inverted CGM can also be thought of as the truncated version of the inverted CSM for sentence length m. Before the inverted CGM unfolds the n-grams representation to a target sentence, a learned translation transformation $\mathbf{T}$ is applied. The reconstructed vector for the source sentence representation is then added in an incremental manner to the corresponding hidden state h_i in the RLM to predict the word f_i in the target language. The issue that is addressed with the CGM model, where the generation of the target words can now incorporates different parts of the reconstructed source sentence representation, is also later improved by (Bahdanau et al., 2015), where they propose attention mechanism to learn soft-alignment between the source and target sentences.

While the model proposed by (Kalchbrenner and Blunsom, 2013) works quite well for rescoring translation hypotheses from SMT system and computing perplexity of reference translations, using CNN as encoder means that the ordering of the words is not preserved. In an almost similar manner to this approach, (Cho et al., 2014) attempt to map source sentence to a fixed vector representation then back to the target sentence, but with two RNNs as encoder and decoder. The encoder RNN reads each word in the source sentence sequentially until it reaches the end of the sequence, which is marked by an end-of-sequence symbol. The hidden state of RNN after completely reading the source sentence is then encoded to a context vector $\mathbf{c}$, which contains the summary of the whole source sentences. Consider source sentence x with length N and target sentence y with length M. The encoder is formulated as follows:

$$h_t = RNN_{enc}(h_{t-1}, \text{emb}(x_t)) \tag{15}$$

$$\mathbf{c} = \tanh(\mathbf{V}_{enc} \cdot h_N) \tag{16}$$

where $\text{emb}(x_t)$ is a continuous representation of the input word at timestep t and $\mathbf{V}_{enc}$ is a learned transformation for the encoder.

On another part, the decoder RNN is a recurrent language model, conditioned on all the previously generated target words and the context vector $\mathbf{c}$. It is computed as follows, where the decoder hidden state is initialized using the context vector:

$$h'_0 = \tanh(\mathbf{V}_{dec} \cdot \mathbf{c}) \tag{17}$$

$$h'_t = RNN_{dec}(h'_{t-1}, \text{emb}(y_{t-1}), \mathbf{c}) \tag{18}$$

where $\mathbf{V}_{dec}$ is a learned transformation for the decoder. The probability distribution of target words is obtained from a softmax function applied to the output of a feedforward neural network that consists of a single intermediate

layer with maxout units (Goodfellow et al., 2013), using the decoder hidden state, context vector and target word generated from the previous timestep as inputs.

For words representation, one-hot encoding is used to distinguish words in the vocabulary, which are then projected twice, yielding a 100-dimensional embedding for each word. Both the encoder and decoder use GRU instead of LSTM as it is easier to compute and implement. The encoder and decoder components of the model are then trained in an end-to-end fashion in order to estimate the conditional probability of the target sentence given the context vector:

$$p(y|x) = \prod_{t=1}^{M} p(y_t|y_{t-1}, y_{t-2}, ..., y_1, \mathbf{c}) \tag{19}$$

In their paper, they focus on integrating the RNN encoder-decoder pair for conventional phrase-based SMT system. The trained RNN encoder-decoder pair is used to rescore phrase pairs between the source and target sentences. This new score is then added to the existing phrase table and used as additional features in the log-linear model for the phrase-based SMT system. Furthermore, the model is also able to produce well-formed phrases in the target language independently without any influence from the actual phrase table.

Seq2Seq Model

The works of (Kalchbrenner and Blunsom, 2013) and (Cho et al., 2014) however focus only on rescoring and not direct translation, although (Cho et al., 2014) mentioned the possibility of replacing phrase table with phrases generated by the RNN encoder-decoder model. (Sutskever et al., 2014) intro-

duced a sequence-to-sequence model, often called "Seq2Seq", which achieved success in producing direct translation from English to French. The Seq2Seq model involves two RNNs as encoder and decoder, similar to (Cho et al., 2014).

The task of the encoder is to process the input sequence (source sentence in this case) sequentially until it reaches the end-of-sentence (<EOS>) symbol, to generate a fixed-dimensional context vector **c** that represents the input sequence. It is also important to note that the encoder of Seq2Seq model processes input sequence in reverse. The idea behind reversing the input sequence is to reduce the "minimal time lag" problem (Hochreiter and Schmidhuber, 1996). By doing this, the distance between the first few words in the source and target languages are now much closer to each other, therefore making it easier for backpropagation to relate the source sentence and the target sentence. This method makes it easier for the decoder to generate the first few words correctly, and in turn improves the probability of generating the correct target translation. The illustration of the input sequence reversal is depicted in Figure 2.

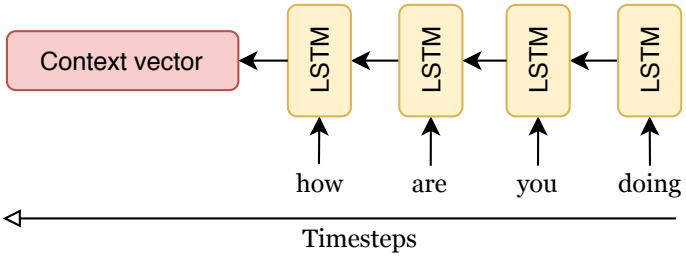


Figure 2: Reversal of tokens in the input sequence

The decoder architecture of the Seq2Seq model is almost similar to the decoder in the RNN encoder-decoder model. It is a recurrent language model, conditioned on all the target words that have been generated until the current timestep and the context vector **c** that encapsulates all the information contained within the source sentence. The main difference between the Seq2Seq

model and the RNN encoder-decoder model is that in Seq2Seq model, the context vector **c** is only used for initializing the decoder hidden state whereas in the RNN encoder-decoder model, **c** is used for decoder hidden state initialization and also for target words generation at all timestep, which is indicated in Eq. 18.

When the source sentence is already encoded into **c** and the decoder is set up with the context vector, a special symbol is passed on to the decoder to mark the generation of the output sequence. In the paper, this is an <EOS> symbol that signifies the end of the input sequence. Afterwards, the decoder will continuously generate words in the target language until it generates the <EOS> symbol. The architecture of Seq2Seq model is depicted in Figure 3.

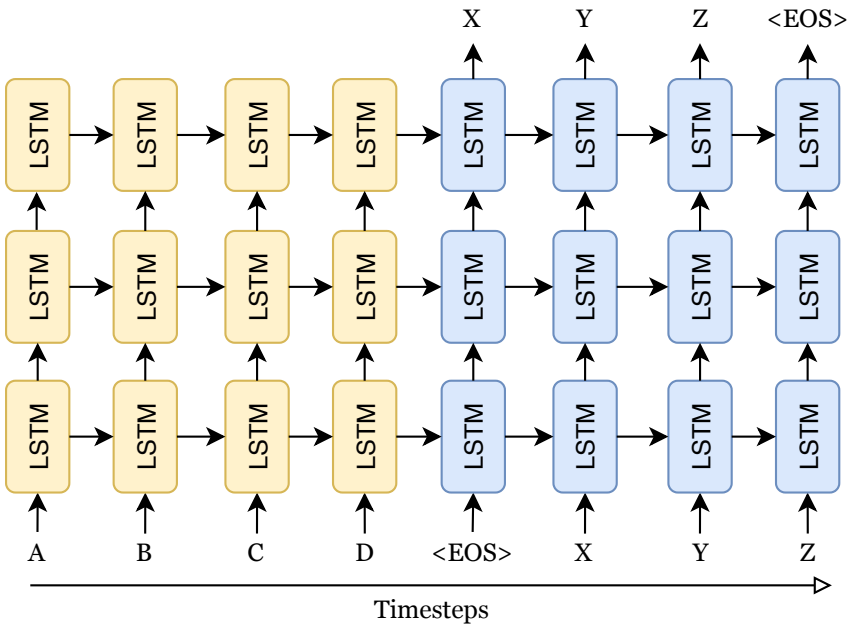


Figure 3: Sequence-to-sequence network

The encoder and decoder architecture of the Seq2Seq model use LSTM as it is capable of learning long-range temporal dependencies. Compared to GRU, LSTM cells consistently perform better in language modeling (Joze-

fowicz et al., 2015) and machine translation tasks (Britz et al., 2017). The performance of LSTM can also be improved further by setting a large bias to the forget gate. To increase the model capacity, the implementation of Seq2Seq uses deep LSTMs (4 layers in the paper) for the encoder and the decoder where the hidden state of encoder final layer at the last timestep is used as context vector $\mathbf{c}$ to initialize the first layer of the decoder LSTM. Each additional layer of the deep LSTMs was also found to be able to reduce the perplexity by 10% (Sutskever et al., 2014).

For Seq2Seq, the goal of the model is to estimate the conditional probability of the output sequence given the context vector:

$$p(y|x) = \prod_{t=1}^{M} p(y_t|\mathbf{c}, y_1, ..., y_{t-1}) \tag{20}$$

where M is output sequence length which may differ from input sequence length. The formulation of the conditional distribution is almost similar to the RNN encoder-decoder model (Eq. 19) but with a minor difference, where here $\mathbf{c}$ is only used to initialize the hidden state of the decoder LSTM and not connected to the decoder at all times. Each distribution $p(y_t|\mathbf{c}, y_1, ..., y_{t-1})$ in Eq. 20 is produced by a softmax operation over all the existing words in the vocabulary.

Both the encoder and decoder LSTMs are trained in an end-to-end approach to maximize the conditional log-likelihood of the correct translation T given the source sentence S, which is defined as the following:

$$1/|\mathcal{S}| \sum_{(T,S)\in\mathcal{S}} \log p(T|S) \tag{21}$$

where $\mathcal{S}$ is the training set. The most likely translation is approximated by using a left-to-right beam search decoder instead of a greedy search decoder,

as greedy decoding only generate most probable word on each timestep which may reduce the overall quality of the predicted translation. During the decoding process, a B number of most probable partial translations (hypotheses) are tracked according to beam size B. Each partial hypothesis is extended on each decoder step with all the words in the vocabulary while discarding all but the B most likely partial hypotheses. When the decoder produces the <EOS> symbol for a hypothesis, it is then removed from the beam search and added to the list of complete hypothesis, while other hypotheses are still explored by the beam search. Additionally, the score of each complete hypothesis is normalized by its length to produce translation with the highest score, due to the bias of the decoding method to select shorter translation.

Seq2Seq is the first model that proves that the performance of a pure neural machine translation system is able to surpass a conventional phrase-based SMT system with a significant margin (Sutskever et al., 2014). The model also surprisingly able to generate high-quality translations even on long sentences, which were initially thought to be difficult due to the limited capacity of LSTM memory cell. Furthermore, vector representations that are produced by the encoder component of the Seq2Seq model are aware of the syntactic and semantic information conveyed in the source language.

Other Aspects of Seq2Seq and Neural Machine Translation

The aforementioned works (Kalchbrenner and Blunsom, 2013; Cho et al., 2014; Sutskever et al., 2014) have drastically changed the field of machine translation. A neural machine translation system offers better performance in term of capturing phrase similarities and long-term dependencies compared to a phrase-based SMT system. In terms of complexity, a NMT system also

needs lesser engineering effort as it requires no feature engineering and applicable to all language pairs without a need for language-specific tailoring. On the other side, NMT approach is less interpretable due to the continuous representations and non-linear properties of neural networks, which causes difficulty in associating hidden states with the language structures and understanding of the translation process.

Seq2Seq, as one of the most promising NMT systems at the time of its inception, also still has some limitations on its own. First of all, the model only learn long-term dependencies in a left-to-right direction, which means that information from words after the current word is not taken into account. This problem can be fixed by using bidirectional RNNs (Schuster and Paliwal, 1997) for the encoder component to read the input sequence from both directions with 2 different RNN cells. The final hidden state from the left-to-right and right-to-left RNNs are concatenated or summed to generate a final context vector, which in turn used to initialize decoder hidden state. Illustration of bidirectional encoder using LSTM cells is depicted in Figure 4.

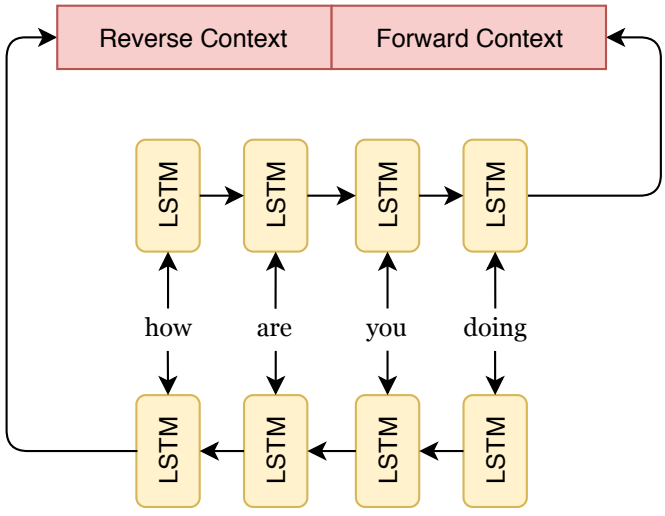


Figure 4: A single-layer bidirectional LSTM encoder network

Another issue with Seq2Seq and NMT systems in general is that the model experiences difficulty in correctly translating rare words because of the limited vocabulary size, which is often attributed to the computationally expensive operation of the softmax function. As a result, a special $<unk>$ symbol is used to represent every out-of-vocabulary (OOV) word during the decoding process. (Luong et al., 2015b) attempt to address this issue by introducing the idea to annotate the training data with alignment information between source and target sentences. The Seq2Seq model is trained on the annotated corpus to keep track of the origins of unknown words that are generated during the translation process, creating links between sentence pairs. The alignment links are used to construct a dictionary, which is used in the post-processing step to replace $<unk>$ symbol with the translation of its corresponding source word. In the case where the translation does not exist in the dictionary, then identity translation is applied to produce the target word.

In the direction of using units smaller than word to solve rare words problem, (Sennrich et al., 2016b) proposed to encode rare and unknown words as sequences of subword units to achieve open-vocabulary translation. Words that appear in the corpus are segmented using an adapted version of Byte Pair Encoding (BPE), where characters are merged instead of bytes. This method initializes the vocabulary with characters and expand the vocabulary content with most frequent character n-grams, which are eventually merged into a single symbol. Subsequently, word segments in the vocabulary can be used to train a NMT system. Compared with word-level approaches, this technique is able to generate unseen words in the training data. This makes it more appropriate for translation tasks, where morphological changes are often required, in particular for languages that use compounding and agglutination widely.

(Luong and Manning, 2016) also proposed a hybrid model using mixture of words and characters, where the translation task is mostly performed at the word level, utilizing the character component to deal with rare words. The hybrid architecture consists of a deep LSTM encoder-decoder model to translate at the word level with two additional character-level LSTMs to compute representations for rare words and recover the unknown target words character-by-character, which are originally represented as $<unk>$. For the encoder part of the model, instead of replacing the rare words with $<unk>$ symbol, a deep character-level LSTM is used to learn over characters of rare source words, where the final hidden state of the LSTM is used as the representation for the current rare word. On the decoder side, another deep LSTM is trained to translate at the character level given the current word context. This character-level decoder is then used whenever the word-level NMT generates an $<unk>$ symbol, to produce the correct form of the unknown target word. By adjusting the vocabulary size, the proportion of word and character-based models in the hybrid architecture can also be controlled to achieve the optimal performance.

The most critical issue with the Seq2Seq model however, is the usage of a single fixed-length context vector. This creates information bottleneck as all the information in the input sequence is encapsulated in the context vector. (Bahdanau et al., 2015) proposed attention mechanism, where instead of using a single fixed-length vector, the input sequence is encoded into a sequence of vectors and directly linked to the decoder, thus allowing the decoder to choose a subset of vectors adaptively while decoding the target sequence. This mechanism also allows a model to handle very long output sequences, as the decoder has access to the input sequence at all timesteps through the connections.

The attention mechanism consists of two stages, which are computing the context vector and the attentional vector. First of all, the hidden state of

the decoder is compared with individual hidden states in the source sequence to learn attention scores using a feedforward neural network. Apart from feedforward neural network, dot-product (Luong et al., 2015a) and scaled dot-product (Vaswani et al., 2017) can also be used as scoring function. These scores reflect the significance of each source token for generating the output and used to compute the context vector as a weighted average of the hidden states from the source sequence. The context vector is then concatenated with the decoder hidden state and passed through another feedforward neural network to obtain the attentional vector, which is used to produce the target word. Illustration of the attention mechanism is shown in Figure 5.

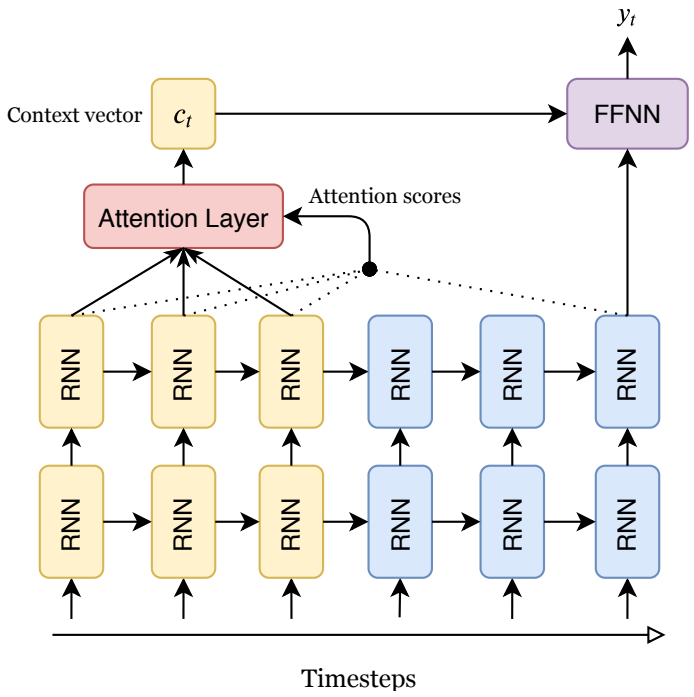


Figure 5: Attention mechanism

Intuitively, the attention mechanism can be perceived as a soft alignment between the words in a source sentence and target translation. It allows the model to search and focus only on important information that is relevant

for the generation of the target word. Furthermore, it is possible to use attention mechanism without any recurrent network units to perform machine translation, using the Transformer architecture (Vaswani et al., 2017).

Seq2Seq NMT system is mainly built for a single language pair, which is not effective for multilingual translation due to the scaling difficulty with regards to the amount of data and training time that are required to support all language pairs. By adding a token to indicate the target language to the source sentence, a single Seq2Seq model is also capable to translate between any two languages (Johnson et al., 2017). The model uses a shared wordpiece model vocabulary (Schuster and Nakajima, 2012) and shared parameters across different languages, allowing implicit bridging (zero-shot translation) between a language pair that has no explicit parallel training data.

NMT systems also rely heavily on large parallel corpora, which are unfortunately sparse for a vast majority of language pairs. (Sennrich et al., 2016a) proposed automatic generation of sentence pairs using back-translation, where given a monolingual corpus for the target sentences, a target-to-source model is trained and applied to the target sentences to generate synthetic source sentences, which in return used for training the source-to-target model. Beside augmenting training sets to improve translation performance, it is possible to use monolingual corpora to perform machine translation in an unsupervised setting (Lample et al., 2018). In the proposed architecture, language models are trained on both source and target languages using BPE (Sennrich et al., 2016b) tokens from the joint monolingual corpora. The language models are then incorporated in a translation system which uses iterative back-translation to couple source-to-target and target-to-source model, effectively converting the unsupervised problem to a supervised learning task.

Conclusion

The Seq2Seq model enables sequence modeling approach that can deal with input and output sequences of varying length separately, optimized as a single system. It is composed of an encoder-decoder neural network, which is a probabilistic conditional generative model of highly dimensional, structured data. The encoder captures the information that is contained in the source sequence, yielding a distributed representation of the source. On the other hand, the decoder takes the distributed representation of the source as well as previously generated context tokens in the target sequence to create a distributed representation and distribution over the next target token in an autoregressive manner.

Fixed-length encodings used in the Seq2Seq model require compressing a large amount of information in the input sequences to a single vector. However, keeping variable-length encodings for the input sequences can prove beneficial, as the model can learn an encoded representation for each source token instead of having to compress all the information (e.g. by using pooling operator such as attention mechanism). Apart from machine translation, Seq2Seq models also proved to be effective for tasks such as text summarization (Nallapati et al., 2016), question answering (Yin et al., 2016), conversational modeling (Vinyals and Le, 2015), and image captioning (Xu et al., 2015; Vinyals et al., 2015).

References

D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. In Y. Bengio and Y. LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1409.0473.

Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, March 1994. ISSN 1045-9227. doi: 10.1109/72.279181.

D. Britz, A. Goldie, M.-T. Luong, and Q. Le. Massive exploration of neural machine translation architectures. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1442–1451, Copenhagen, Denmark, Sept. 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1151. URL https://www.aclweb.org/anthology/D17-1151.

K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, Oct. 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1179. URL https://www.aclweb.org/anthology/D14-1179.

F. A. Gers and J. Schmidhuber. Recurrent nets that time and count. In *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks, IJCNN 2000, Neural Computing: New Challenges and Perspectives for the New Millennium, Como, Italy, July 24-27, 2000, Volume 3*, pages 189–194. IEEE Computer Society, 2000. doi: 10.1109/IJCNN.2000.861302. URL https://doi.org/10.1109/IJCNN.2000.861302.

I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio. Maxout networks. In *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ICML'13, pages III–1319–III–1327. JMLR.org, 2013. URL http://dl.acm.org/citation.cfm?id=3042817.3043084.

E. Grefenstette, M. Sadrzadeh, S. Clark, B. Coecke, and S. Pulman. Concrete sentence spaces for compositional distributional models of meaning. In *Proceedings of the Ninth International Conference on Computational Semantics*, IWCS '11, pages 125–134, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=2002669.2002683.

G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, Nov 2012. ISSN 1053-5888. doi: 10.1109/MSP.2012.2205597.

S. Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 6(2):107–116, Apr. 1998. ISSN 0218-4885. doi: 10.1142/S0218488598000094. URL http://dx.doi.org/10.1142/S0218488598000094.

S. Hochreiter and J. Schmidhuber. Lstm can solve hard long time lag problems. In *Proceedings of the 9th International Conference on Neural Information Processing Systems*, NIPS'96, pages 473–479, Cambridge, MA, USA, 1996. MIT Press. URL http://dl.acm.org/citation.cfm?id=2998981.2999048.

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Com-*

put., 9(8):1735–1780, Nov. 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL http://dx.doi.org/10.1162/neco.1997.9.8.1735.

M. Johnson, M. Schuster, Q. V. Le, M. Krikun, Y. Wu, Z. Chen, N. Thorat, F. Viégas, M. Wattenberg, G. Corrado, M. Hughes, and J. Dean. Google's multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics*, 5: 339–351, 2017. doi: 10.1162/tacl_a_00065. URL https://www.aclweb.org/anthology/Q17-1024.

R. Jozefowicz, W. Zaremba, and I. Sutskever. An empirical exploration of recurrent network architectures. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 2342–2350. JMLR.org, 2015. URL http://dl.acm.org/citation.cfm?id=3045118.3045367.

N. Kalchbrenner and P. Blunsom. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709, Seattle, Washington, USA, Oct. 2013. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/D13-1176.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, pages 1097–1105, USA, 2012. Curran Associates Inc. URL http://dl.acm.org/citation.cfm?id=2999134.2999257.

G. Lample, M. Ott, A. Conneau, L. Denoyer, and M. Ranzato. Phrase-based & neural unsupervised machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5039–5049, Brussels, Belgium, Oct.-Nov. 2018. Associa-

tion for Computational Linguistics. doi: 10.18653/v1/D18-1549. URL https://www.aclweb.org/anthology/D18-1549.

M.-T. Luong and C. D. Manning. Achieving open vocabulary neural machine translation with hybrid word-character models. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1054–1063, Berlin, Germany, Aug. 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1100. URL https://www.aclweb.org/anthology/P16-1100.

T. Luong, H. Pham, and C. D. Manning. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal, Sept. 2015a. Association for Computational Linguistics. doi: 10.18653/v1/D15-1166. URL https://www.aclweb.org/anthology/D15-1166.

T. Luong, I. Sutskever, Q. Le, O. Vinyals, and W. Zaremba. Addressing the rare word problem in neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 11–19, Beijing, China, July 2015b. Association for Computational Linguistics. doi: 10.3115/v1/P15-1002. URL https://www.aclweb.org/anthology/P15-1002.

T. Mikolov, M. Karafiát, L. Burget, J. Cernocký, and S. Khudanpur. Recurrent neural network based language model. In T. Kobayashi, K. Hirose, and S. Nakamura, editors, *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*, pages 1045–1048. ISCA, 2010. URL http://www.isca-speech.org/archive/interspeech_2010/i10_1045.html.

T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. In Y. Bengio and Y. LeCun, editors, *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, 2013. URL http://arxiv.org/abs/1301.3781.

R. Nallapati, B. Zhou, C. dos Santos, Ç. GuÌ‡lçehre, and B. Xiang. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290, Berlin, Germany, Aug. 2016. Association for Computational Linguistics. doi: 10.18653/v1/K16-1028. URL https://www.aclweb.org/anthology/K16-1028.

J. Pennington, R. Socher, and C. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, Oct. 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1162. URL https://www.aclweb.org/anthology/D14-1162.

M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. Deep contextualized word representations. In *Proc. of NAACL*, 2018.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning Representations by Back-propagating Errors. *Nature*, 323(6088):533–536, 1986. doi: 10.1038/323533a0. URL http://www.nature.com/articles/323533a0.

M. Schuster and K. Nakajima. Japanese and korean voice search. In *International Conference on Acoustics, Speech and Signal Processing*, pages 5149–5152, 2012.

M. Schuster and K. Paliwal. Bidirectional recurrent neural networks. *Trans.*

Sig. Proc., 45(11):2673–2681, Nov. 1997. ISSN 1053-587X. doi: 10.1109/78.650093. URL http://dx.doi.org/10.1109/78.650093.

R. Sennrich, B. Haddow, and A. Birch. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96, Berlin, Germany, Aug. 2016a. Association for Computational Linguistics. doi: 10.18653/v1/P16-1009. URL https://www.aclweb.org/anthology/P16-1009.

R. Sennrich, B. Haddow, and A. Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany, Aug. 2016b. Association for Computational Linguistics. doi: 10.18653/v1/P16-1162. URL https://www.aclweb.org/anthology/P16-1162.

R. Socher, B. Huval, C. D. Manning, and A. Y. Ng. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 1201–1211, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=2390948.2391084.

I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, pages 3104–3112, Cambridge, MA, USA, 2014. MIT Press. URL http://dl.acm.org/citation.cfm?id=2969033.2969173.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin. Attention is all you need. In I. Guyon,

U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc., 2017. URL http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf.

O. Vinyals and Q. V. Le. A neural conversational model. *CoRR*, abs/1506.05869, 2015. URL http://arxiv.org/abs/1506.05869.

O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. Show and tell: A neural image caption generator. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3156–3164, June 2015. doi: 10.1109/CVPR.2015.7298935.

K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. In F. Bach and D. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 2048–2057, Lille, France, 07–09 Jul 2015. PMLR. URL http://proceedings.mlr.press/v37/xuc15.html.

J. Yin, X. Jiang, Z. Lu, L. Shang, H. Li, and X. Li. Neural generative question answering. In *Proceedings of the Workshop on Human-Computer Question Answering*, pages 36–42, San Diego, California, June 2016. Association for Computational Linguistics. doi: 10.18653/v1/W16-0106. URL https://www.aclweb.org/anthology/W16-0106.