April 01, 2019

Sequence Transduction Neural Networks With Localized Self-Attention

Nan Ding

Radu Soricut

Sequence Transduction Neural Networks With Localized Self-Attention

BACKGROUND

This document relates to transducing sequences using neural networks. Neural networks are machine learning models that employ one or more layers of nonlinear units to predict an output for a received input. Some neural networks include one or more hidden layers in addition to an output layer. The output of each hidden layer is used as input to the next layer in the network, i.e., the next hidden layer or the output layer. Each layer of the network generates an output from a received input in accordance with current values of a respective set of parameters.

SUMMARY

This document describes technologies for automatically generating a target sequence that includes a respective output at each of multiple positions in an output order from an input sequence that includes a respective input at each of multiple positions in an input order, i.e., transduces the input sequence into the output sequence. In particular, this document describes a system that generates the output sequence using an encoder neural network and a decoder neural network that are both attention-based. The encoder neural network includes one or more localized self-attention modules while the decoder includes one or more self-attention modules and also one or more encoder-decoder attention modules. In some cases, the encoder includes only localized self-attention modules, i.e., does not include any (non-localized) self-attention modules.

Because the encoder and the decoder are attention-based, the sequence transduction neural network can transduce sequences more accurately than existing networks that are based

on convolutional layers or recurrent layers. Additionally, the use of an attention-based architecture allows the sequence transduction neural network to transduce sequences quicker, to be trained faster, or both, e.g., because the operation of the network can be more easily parallelized.

More particularly, because the encoder is based on localized self-attention, the sequence transduction neural network can to transduce sequences quicker, to be trained faster, or both, than even neural networks that employ self-attention in the encoder while still achieving comparable or better results to those neural networks.

Therefore, the neural networks described in this document can consume fewer computational resources during inference and training than existing approaches.

DETAILED DESCRIPTION

This description describes a system implemented as computer programs on one or more computers in one or more locations that generates a target sequence that includes a respective output at each of multiple positions in an output order from an input sequence that includes a respective input at each of multiple positions in an input order, i.e., transduces the input sequence into the target sequence.

For example, the system may be a neural machine translation system. That is, if the input sequence is a sequence of words in an original language, e.g., a sentence or phrase, the target sequence may be a translation of the input sequence into a target language, i.e., a sequence of words in the target language that represents the sequence of words in the original language.

As another example, the system may be a speech recognition system. That is, if the input sequence is a sequence of audio data representing a spoken utterance, the target sequence

may be a sequence of graphemes, characters, or words that represents the utterance, i.e., is a transcription of the input sequence.

As another example, the system may be a natural language processing system. For example, if the input sequence is a sequence of words in an original language, e.g., a sentence or phrase, the target sequence may be a summary of the input sequence in the original language, i.e., a sequence that has fewer words than the input sequence but that retains the essential meaning of the input sequence. The words in the summary can include words that are not in the original sequence. For example, the original sequence can be words from an electronic document, e.g., a news article, a blog post, an encyclopedia entry, and the summary can be title, headline, or bullet-pointed summary of the electronic document. As another example, if the input sequence is a sequence of words that form a question, the target sequence can be a sequence of words that form an answer to the question.

As another example, the system may be part of a computer-assisted medical diagnosis system. For example, the input sequence can be a sequence of data from an electronic medical record and the target sequence can be a sequence of predicted treatments.

As another example, the system may be part of an image processing system. For example, the input sequence can be an image, i.e., a sequence of color values from the image, and the output can be a sequence of text that describes the image. As another example, the input sequence can be a sequence of text or a different context and the output sequence can be an image that describes the context.

In some cases, the system can pre-process the sequences before they are provided as input to the sequence transduction neural network and post-process the sequences after they are generated as outputs by the sequence transduction neural network. For example, when the input

and output sequence are sequences of words, the system can first tokenize the sequence, e.g., into characters, word pieces, or using a sub-word characterization model. The neural network can then generate a sequence of output tokens, and the system can post-process the generated sequence to generate the final output sequence of words from the sequence of tokens.

FIG. 1 shows an example neural network system 100 for generating a target sequence from an input sequence. The neural network system 100 is an example of a system implemented as computer programs on one or more computers in one or more locations, in which the systems, components, and techniques described below can be implemented.

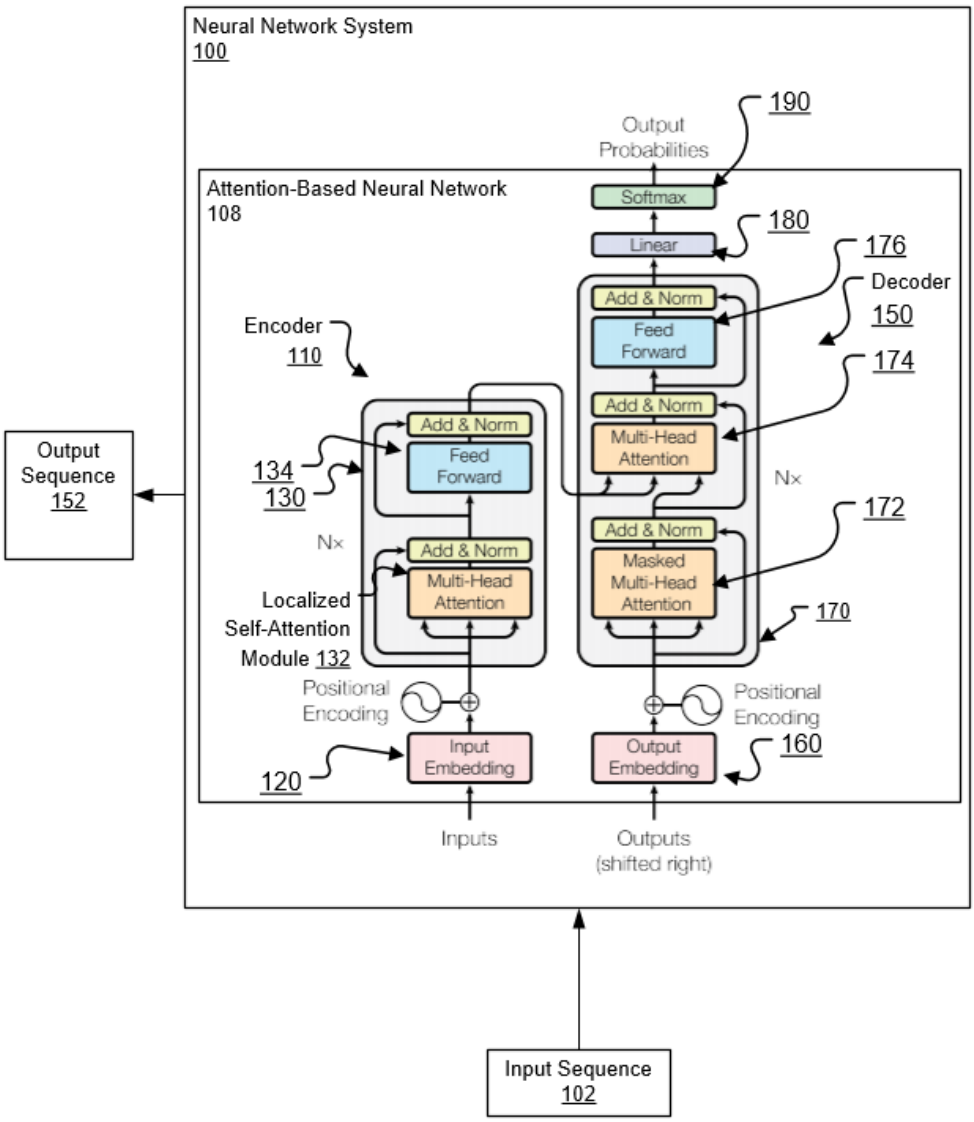

FIG. 1

The neural network system 100 receives an input sequence 102 and processes the input sequence 102 to transduce the input sequence 102 into an output sequence 152. The input sequence 102 has a respective network input at each of multiple input positions in an input order and the output sequence 152 has a respective network output at each of multiple output positions in an output order. That is, the input sequence 102 has multiple inputs arranged according to an input order and the output sequence 152 has multiple outputs arranged according to an output order.

As described above, the neural network system 100 can perform any of a variety of tasks that require processing sequential inputs to generate sequential outputs. To perform these tasks, the neural network system 100 includes an attention-based sequence transduction neural network 108, which in turn includes an encoder neural network 110 and a decoder neural network 150.

The encoder neural network 110 is configured to receive the input sequence 102 and generate a respective encoded representation of each of the network inputs in the input sequence. Generally, an encoded representation is a vector or other ordered collection of numeric values. The decoder neural network 150 is then configured to use the encoded representations of the network inputs to generate the output sequence 152.

Generally, both the encoder 110 and the decoder 150 are attention-based. In some cases, neither the encoder nor the decoder include any convolutional layers or any recurrent layers. As will be described in more detail below, the encoder 110 includes one or more localized self-attention modules while the decoder 150 includes one or more self-attention modules and also one or more encoder-decoder attention modules. In some cases, the encoder

includes only localized self-attention modules, i.e., does not include any (non-localized) self-attention modules.

In particular, the encoder neural network 110 includes an embedding layer 120 and a sequence of one or more encoder subnetworks 130. In particular, as shown in FIG. 1, the encoder neural network includes N encoder subnetworks 130.

The embedding layer 120 is configured to, for each network input in the input sequence, map the network input to a numeric representation of the network input in an embedding space, e.g., into a vector in the embedding space. The embedding layer 120 then provides the numeric representations of the network inputs to the first subnetwork in the sequence of encoder subnetworks 130, i.e., to the first encoder subnetwork 130 of the N encoder subnetworks 130.

In particular, in some implementations, the embedding layer 120 is configured to map each network input to an embedded representation of the network input and then combine, e.g., sum or average, the embedded representation of the network input with a positional embedding of the input position of the network input in the input order to generate a combined embedded representation of the network input. That is, each position in the input sequence has a corresponding embedding and for each network input the embedding layer 120 combines the embedded representation of the network input with the embedding of the network input's position in the input sequence. Such positional embeddings can enable the model to make full use of the order of the input sequence without relying on recurrence or convolutions.

In some cases, the positional embeddings are learned during the training of the sequence transduction neural network 108. In some other cases, the positional embeddings are fixed and are different for each position. For example, the embeddings can be made up of sine

and cosine functions of different frequencies. The use of sinusoidal positional embeddings may allow the model to extrapolate to longer sequence lengths, which can increase the range of applications for which the model can be employed.

The combined embedded representation is then used as the numeric representation of the network input.

Each of the encoder subnetworks 130 is configured to receive a respective encoder subnetwork input for each of the plurality of input positions and to generate a respective subnetwork output for each of the plurality of input positions. The encoder subnetwork outputs generated by the last encoder subnetwork in the sequence are then used as the encoded representations of the network inputs.

For the first encoder subnetwork in the sequence, the encoder subnetwork input is the numeric representations generated by the embedding layer 120, and, for each encoder subnetwork other than the first encoder subnetwork in the sequence, the encoder subnetwork input is the encoder subnetwork output of the preceding encoder subnetwork in the sequence.

Each encoder subnetwork 130 includes an encoder localized self-attention module 132. A localized self-attention module, in contrast with a self-attention module, is a module that, for a given position in the input order, attends to less than all of the positions in the input order when generating an output for the given position. More specifically, for a particular input position, a localized self-attention module operates only on data that corresponds to positions that are within a window of fixed size of the particular input position in the input order. The window can be, e.g., one, two, five, or ten positions but is always less than the total number of inputs in the input sequence. In other words, if the window size is a fixed integer d, when performing the processing for a particular input position, a localized self-attention module

operates on data corresponding to the d positions before the input position in the input order, data at the input position, and d positions after the input position in the input order. For example, when the window size is one position, the localized self-attention module corresponding to the particular position operates on data at the position before the particular position, on data at the particular position, and on data at the position after the particular position. When the window size is ten positions, the localized self-attention module corresponding to the particular position operates on data at the ten positions before the particular position, on data at the particular position, and on data at the ten positions after the particular position.

A self-attention module, on the other hand, operates on data from all of the input positions in the input order when generating the output for any given position in the sequence.

Because localized self-attention modules operate on less data, i.e., data from fewer positions, computations performed by localized self-attention modules are less computationally expensive than those performed by self-attention modules. When the window size is significantly less than the total size of the sequence and the encoder includes multiple modules, the decrease in computational cost, e.g., in the amount of memory and processing power used by the encoder, can be significant.

Localized self-attention modules also differ from masked self-attention modules. A masked self-attention module corresponding to a particular position operates only on data from positions before the particular position in the order. A localized self-attention module, on the other hand, operates on data that is both before and after the particular position in the order and does not operate on all of the data that is before the particular position in the order.

As shown in FIG. 1, the encoder localized self-attention module 132 is configured to receive the subnetwork input for each of the plurality of input positions. For each particular

input position in the input order, the encoder localized self-attention module 132 applies a localized self-attention mechanism over the encoder subnetwork inputs at input positions within a window of a fixed size of the particular input position to generate a respective output for the particular input position. In some cases, the localized self-attention mechanism is a multi-head attention mechanism.

To apply a localized self-attention mechanism over the encoder subnetwork inputs at input positions within the window, the encoder localized self-attention module 132 includes a plurality of encoder localized self-attention layers. Each encoder localized self-attention layer is configured to: apply a learned query linear transformation to each encoder subnetwork input at each input position to generate a respective query for each input position, apply a learned key linear transformation to each encoder subnetwork input at each input position to generate a respective key for each input position, and apply a learned value linear transformation to each encoder subnetwork input at each input position to generate a respective value for each input position. For each input position, each encoder localized self-attention layer is configured to determine a respective input-position specific weight for each of the input positions by applying a comparison function between the query for the input position and the keys for positions within the window of the input position, and determine an initial encoder self-attention output for the input position by determining a weighted sum of the values for the positions within the window of the input position weighted by the corresponding input-position specific weights for the input positions.

The encoder localized self-attention module 132 is then configured to, for each input position, combine the initial encoder self-attention outputs for the input position generated by the encoder self-attention layers to generate the output for the encoder localized self-attention

module 132. In some implementations, the encoder localized self-attention layers can operate in parallel.

In some implementations, each of the encoder subnetworks 130 also includes a residual connection layer that combines the outputs of the encoder self-attention module with the inputs to the encoder self-attention module to generate an encoder self-attention residual output and a layer normalization layer that applies layer normalization to the encoder self-attention residual output. These two layers are collectively referred to as an "Add & Norm" operation in FIG. 1.

Some or all of the encoder subnetworks can also include a position-wise feed-forward layer 134 that is configured to operate on each position in the input sequence separately. In particular, for each input position, the feed-forward layer 134 is configured receive an input at the input position and apply a sequence of transformations to the input at the input position to generate an output for the input position. For example, the sequence of transformations can include two or more learned linear transformations each separated by an activation function, e.g., a non-linear elementwise activation function, e.g., a ReLU activation function, which can allow for faster and more effective training on large and complex datasets. The inputs received by the position-wise feed-forward layer 134 can be the outputs of the layer normalization layer when the residual and layer normalization layers are included or the outputs of the encoder self-attention module 132 when the residual and layer normalization layers are not included. The transformations applied by the layer 134 will generally be the same for each input position (but different feed-forward layers in different subnetworks will apply different transformations).

In cases where an encoder subnetwork 130 includes a position-wise feed-forward layer 134, the encoder subnetwork can also include a residual connection layer that combines the

outputs of the position-wise feed-forward layer with the inputs to the position-wise feed-forward layer to generate an encoder position-wise residual output and a layer normalization layer that applies layer normalization to the encoder position-wise residual output. These two layers are also collectively referred to as an "Add & Norm" operation in FIG. 1. The outputs of this layer normalization layer can then be used as the outputs of the encoder subnetwork 130.

Once the encoder neural network 110 has generated the encoded representations, the decoder neural network 150 is configured to generate the output sequence in an auto-regressive manner.

That is, the decoder neural network 150 generates the output sequence, by at each of a plurality of generation time steps, generating a network output for a corresponding output position conditioned on (i) the encoded representations and (ii) network outputs at output positions preceding the output position in the output order.

In particular, for a given output position, the decoder neural network generates an output that defines a probability distribution over possible network outputs at the given output position. The decoder neural network can then select a network output for the output position by sampling from the probability distribution or by selecting the network output with the highest probability.

Because the decoder neural network 150 is auto-regressive, at each generation time step, the decoder 150 operates on the network outputs that have already been generated before the generation time step, i.e., the network outputs at output positions preceding the corresponding output position in the output order. In some implementations, to ensure this is the case during both inference and training, at each generation time step the decoder neural network 150 shifts the already generated network outputs right by one output order position (i.e.,

introduces a one position offset into the already generated network output sequence) and (as will be described in more detail below) masks certain operations so that positions can only attend to positions up to and including that position in the output sequence (and not subsequent positions). While the remainder of the description below describes that, when generating a given output at a given output position, various components of the decoder 150 operate on data at output positions preceding the given output positions (and not on data at any other output positions), it will be understood that this type of conditioning can be effectively implemented using the shifting described above.

The decoder neural network 150 includes an embedding layer 160, a sequence of decoder subnetworks 170, a linear layer 180, and a softmax layer 190. In particular, as shown in FIG. 1, the decoder neural network includes N decoder subnetworks 170. However, while the example of FIG. 1 shows the encoder 110 and the decoder 150 including the same number of subnetworks, in some cases the encoder 110 and the decoder 150 include different numbers of subnetworks. That is, the decoder 150 can include more or fewer subnetworks than the encoder 110.

The embedding layer 160 is configured to, at each generation time step, for each network output at an output position that precedes the current output position in the output order, map the network output to a numeric representation of the network output in the embedding space. The embedding layer 160 then provides the numeric representations of the network outputs to the first subnetwork 170 in the sequence of decoder subnetworks, i.e., to the first decoder subnetwork 170 of the N decoder subnetworks.

In particular, in some implementations, the embedding layer 160 is configured to map each network output to an embedded representation of the network output and combine the

embedded representation of the network output with a positional embedding of the output position of the network output in the output order to generate a combined embedded representation of the network output. The combined embedded representation is then used as the numeric representation of the network output. The embedding layer 160 generates the combined embedded representation in the same manner as described above with reference to the embedding layer 120.

Each decoder subnetwork 170 is configured to, at each generation time step, receive a respective decoder subnetwork input for each of the plurality of output positions preceding the corresponding output position and to generate a respective decoder subnetwork output for each of the plurality of output positions preceding the corresponding output position (or equivalently, when the output sequence has been shifted right, each network output at a position up to and including the current output position).

In particular, each decoder subnetwork 170 includes two different attention modules: a decoder self-attention module 172 and an encoder-decoder attention module 174.

Each decoder self-attention module 172 is configured to, at each generation time step, receive an input for each output position preceding the corresponding output position and, for each of the particular output positions, apply an attention mechanism over the inputs at the output positions preceding the corresponding position using one or more queries derived from the input at the particular output position to generate a updated representation for the particular output position. That is, the decoder self-attention module 172 applies an attention mechanism that is masked so that it does not attend over or otherwise process any data that is not at a position preceding the current output position in the output sequence.

Each encoder-decoder attention module 174, on the other hand, is configured to, at each generation time step, receive an input for each output position preceding the corresponding output position and, for each of the output positions, apply an attention mechanism over the encoded representations at the input positions using one or more queries derived from the input for the output position to generate an updated representation for the output position. Thus, the encoder-decoder attention module 174 applies attention over encoded representations while the encoder self-attention module 172 applies attention over inputs at output positions.

In FIG. 1, the decoder self-attention module 172 is shown as being before the encoder-decoder attention module in the processing order within the decoder subnetwork 170. In other examples, however, the decoder self-attention module 172 may be after the encoder-decoder attention module 174 in the processing order within the decoder subnetwork 170 or different subnetworks may have different processing orders.

In some implementations, each decoder subnetwork 170 includes, after the decoder self-attention module 172, after the encoder-decoder attention module 174, or after each of the two modules, a residual connection layer that combines the outputs of the attention module with the inputs to the attention module to generate a residual output and a layer normalization layer that applies layer normalization to the residual output. FIG. 1 shows these two layers being inserted after each of the two modules, both referred to as an "Add & Norm" operation.

Some or all of the decoder subnetwork 170 also include a position-wise feed-forward layer 176 that is configured to operate in a similar manner as the position-wise feed-forward layer 134 from the encoder 110. In particular, the layer 176 is configured to, at each generation time step: for each output position preceding the corresponding output position: receive an input at the output position, and apply a sequence of transformations to the input at the output position

to generate an output for the output position. For example, the sequence of transformations can include two or more learned linear transformations each separated by an activation function, e.g., a non-linear elementwise activation function, e.g., a ReLU activation function. The inputs received by the position-wise feed-forward layer 176 can be the outputs of the layer normalization layer (following the last attention module in the subnetwork 170) when the residual and layer normalization layers are included or the outputs of the last attention module in the subnetwork 170 when the residual and layer normalization layers are not included.

In cases where a decoder subnetwork 170 includes a position-wise feed-forward layer 176, the decoder subnetwork can also include a residual connection layer that combines the outputs of the position-wise feed-forward layer with the inputs to the position-wise feed-forward layer to generate a decoder position-wise residual output and a layer normalization layer that applies layer normalization to the decoder position-wise residual output. These two layers are also collectively referred to as an "Add & Norm" operation in FIG. 1. The outputs of this layer normalization layer can then be used as the outputs of the decoder subnetwork 170.

At each generation time step, the linear layer 180 applies a learned linear transformation to the output of the last decoder subnetwork 170 in order to project the output of the last decoder subnetwork 170 into the appropriate space for processing by the softmax layer 190. The softmax layer 190 then applies a softmax function over the outputs of the linear layer 180 to generate the probability distribution over the possible network outputs at the generation time step. As described above, the decoder 150 can then select a network output from the possible network outputs using the probability distribution.

ABSTRACT

A system for transducing an input sequence into a target sequence is described. The system includes a sequence transduction neural network for transducing an input sequence having a respective network input at each of a plurality of input positions in an input order into an output sequence having a respective network output at each of a plurality of output positions in an output order. The sequence transduction neural network includes an encoder neural network and a decoder neural network. The encoder neural network is configured to receive the input sequence and generate a respective encoded representation of each of the network inputs in the input sequence. The encoder neural network includes a sequence of one or more encoder subnetworks, in which each encoder subnetwork is configured to receive a respective encoder subnetwork input for each of the plurality of input positions and to generate a respective encoder subnetwork output for each of the plurality of input positions. Each encoder subnetwork includes an encoder localized self-attention module that is configured to receive the subnetwork input for each of the plurality of input positions and, for each particular input position in the input order, the encoder localized self-attention module is configured to apply a localized self-attention mechanism over the encoder subnetwork inputs at input positions within a window of a fixed size of the particular input position to generate a respective output for the particular input position. The decoder neural network is configured to receive the encoded representations and generate the output sequence.