Is attention really all you need?

Kyle Roth
Brigham Young University
kylrth@gmail.com

Contents

Abstract

This is a wonderful abstract.

1 Introduction

The attention mechanism is an exciting development in the artificial intelligence community. The concept is simple, and rooted in our understanding of attention in biological intelligence [Larochelle and Hinton, 2010; Hassabis *et al.*, 2017]. Attention mechanisms also lend themselves to more intuitive interpretation, an attractive feature as concerns about model interpretability gain traction. In this review, we observe the trajectory that current research is taking with regard to the theory and application of attention, and suggest how future work should try to understand why attention is so useful for such varied applications.

1.1 Background on neural machine learning

Machine learning is a branch of artificial intelligence that deals with training a model to perform tasks by providing the model with data. There are various types of machine learning algorithms, applicable to different tasks. One of the most popular algorithms is the neural network. A neural network is a system of connections between simple functions (e.g. multiplication, summation, ReLU [Nair and Hinton, 2010]), which together approximate a larger, more complex function. It has been proven that sufficiently large neural networks can approximate arbitrary continuous functions [Cybenko, 1989]. Theoretically, this means that for every reasonable mapping there exists a neural network that replicates it. Some example mappings are listed in Table 1.

Input	Output
images	the objects the image contains
audio recordings	a text transcription
English text	a French translation

Table 1: Mappings that a neural network can theoretically learn. Mappings must be consistent; that is, each input should only have one corresponding output.

Most commonly, neural networks undergo supervised training, meaning that they are shown the correct output for each training example and made to improve in some way. Neural networks generally use backpropagation[1] and gradient descent[2] to train weights that are present in the simple functions of the network. Backpropagation uses the difference between the correct example and the network's output to determine how the weights in the network need to change. Figure 1 provides a visualization of backpropagation in a neural network.

While the theoretical limitations of neural networks are quite permissive, it is relatively difficult to train them. Backpropagation is sensitive to "noisy" or incorrect data, meaning that data cleaning and preparation is at least as important as architecture selection. The training process is also relatively expensive in terms of computation time and memory usage; neural networks are commonly trained for days or

[1] *Backpropagation* refers to the calculation of the gradient for each trainable weight in the network. See [Hecht-Nielsen, 1992] for details on backpropagation.

[2] *Gradient descent* is a method of training wherein the weights are updated in the direction opposite the gradient at each training step [Ruder, 2016].

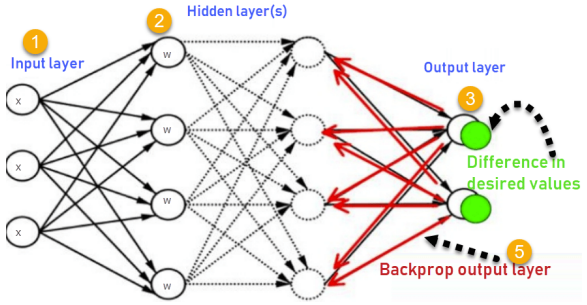

Figure 1: A visualization showing that backpropagation uses the difference between the desired and actual output to train the weights of the neural network. Image source [Guru99, 2019].

even months [Vaswani *et al.*, 2017].

Another real issue is that backpropagation is unlikely to find the "global optimum", the absolutely optimal set of weights that approximate the function. This is due to the fact that the algorithm only adjusts the weights incrementally, and can get stuck in a "local optimum", a condition where no slight change would improve performance but performance is still not as good as at the global optimum.

These difficulties with training neural networks are the reason that so much effort is given to developing better network architectures. Certain architectures and data representations lend themselves to a smoother search space for the weights, increasing the likelihood that backpropagation reaches a better optimum.

1.2 Description of attention

Attention appears to provide that smoother search space for training the network because its output can explicitly depend on each input given to the network, while also giving an intuitive view of the inputs that the network learns to "attend" to.

Bahdanau *et al.* [2015] first described attention as an "alignment model" for words in a sentence to be translated. They used an RNN encoder-decoder model as others had before [Cho *et al.*, 2014], but for every position in the target sentence, the alignment model produced a "soft alignment"[3] between the target location and every word in the source sentence. See Figure 2 for more details on the model.

The key development here was the use of a soft alignment, meaning that each pair of source and target words was given an alignment score between 0 and 1. This made the alignment calculation differentiable, allowing the alignment to be performed by a feed-forward[4] neural network trained along with the rest of the model.

[3] In machine translation, alignment refers to the pairing of words or phrases in the source sentence with words or phrases in the target sentence.

[4] "Feed-forward" refers to the fact that all layers of the network produce output in the forward direction (as opposed to an RNN for example). An example representation can be seen in Figure 1.

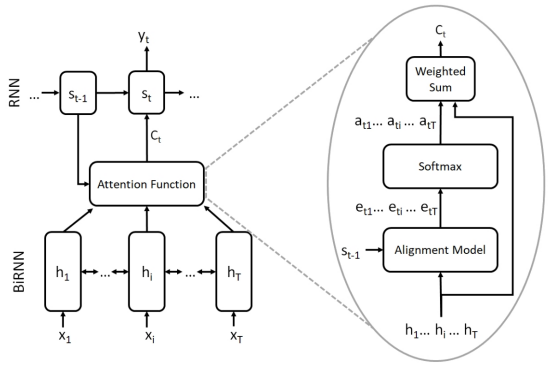

Figure 2: The model used by [Bahdanau *et al.*, 2015]. They used a bidirectional RNN to encode each word x_i into a fixed-length vector h_i. Then, for every position t in the target sentence, the alignment model produced an alignment vector a_t, a measure of alignment between the target location and every word in the source sentence. a_t was used to find the weighted sum $c_t = \sum_{i=1}^{T} a_{ti} h_i$, which was used as the input to the decoder (a unidirectional RNN). Note that the alignment model also received the previous decoder RNN state s_{t-1} as input. Image source [Galassi *et al.*, 2019].

2 Theoretical generalization

Ironically, attention mechanisms have received lots of attention since their conception in 2015. Initially formulated as a word alignment mechanism for text translation [Bahdanau *et al.*, 2015], attention can be generalized in two ways: first, as a mechanism that learns the relationship between variables, and second as a mapping from queries to outputs given a set of key-value pairs.

2.1 Active memory

One of the first applications of attention was in creating a completely differentiable neural Turing machine [Graves *et al.*, 2014]. The model was given memory to read and write from at each input, and an attention mechanism allowed those operations to be differentiable. Just as vanilla neural networks have been shown to approximate arbitrary functions, the neural Turing machine can approximate arbitrary programs.

One of the shortcomings of the neural Turing machine is that the attention mechanism is biased toward attending to one value more than any other [Kaiser and Bengio, 2016]. Thus it is difficult to train the network to perform tasks requiring the model to read from multiple entries in memory, like adding two numbers. Kaiser and Bengio [2016] suggested that attention suffers exactly from this, and offer a mechanism called active memory that attends to all inputs equally. They called their model the extended neural GPU, and achieved promising results with better performance on translating longer sentences.

This appeared to be a promising generalization of attention, but the performance of the extended neural GPU was outstripped by later attention models that solved the single-focus problem using multi-headed attention [Vaswani *et al.*, 2017] (see section 3.2).

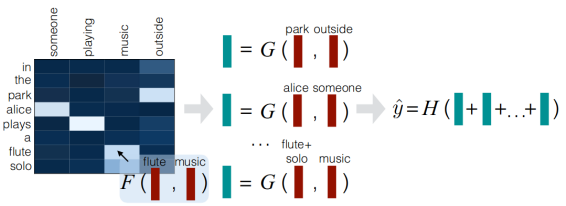

Figure 3: The model used by [Parikh *et al.*, 2016]. F is the attention model used to "decompose the problem into the comparison of aligned subphrases". G compares the aligned subphrases. Both F and G were feed-forward networks in the work.

2.2 Attention as relationships

There is a wide range of relationships that attention mechanisms can model. Specifically, we will see that attention has been generalized to include relationships between two input sequences, relationships within a single sequence, and relationships between arbitrary graphs. This generalization leads us to a view of attention beyond the scope of the word "attention" itself.

Inter-sequence attention

Instead of applying attention to elements of an input sequence to predict an output sequence, Parikh *et al.* [2016] successfully used an attention mechanism to produce relationships between elements of two input sequences. They successfully used this mechanism to perform natural language inference[5]. Figure 3 makes it obvious that the attention function F is determining the strength of the relationship between its two input words; that strength is used by the model to determine the entailment relationship of the pair of sentences.

Self-attention

Parikh *et al.* [2016] also suggest using what they term "intra-sentence attention" to represent relationships between words within each input sentence. This is more broadly referred to as self-attention [Lin *et al.*, 2017; Vaswani *et al.*, 2017]. As the name suggests, self-attentive mechanisms learn the relationship between elements of a single sequence. While RNNs are theoretically able to do this, they struggle with long-distance relationships because they encode all relationships into a single context vector [Bahdanau *et al.*, 2015]. Self-attention overcomes this difficulty.

Attention on arbitrary graphs

Attention can also be used on more complex structures than sequences. Kim *et al.* [2017] show that attention can be generalized to represent relationships between elements of arbitrary graphs[6]. This allows the model to accept parse trees and other useful structures that explicitly encode relationships in the data.

[5]Natural language inference is the task of determining whether the meaning of a sentence entails (implies) or contradicts the meaning of another.

[6]A *graph* in this context refers to a set of elements with connections drawn between them. A sequence is a graph with elements connected one after the other.

This use of attention is relevant because it blurs the line between data and relationships. In this case, some known relationships were provided to the model. When the model was provided with tree structure representations of sentences, Kim *et al.* [2017] observed that the model learned to attend to entire subtrees as well as individual words. Thus the structure introduced in the input served as a way for the attention mechanism to

2.3 Attention as a lookup mapping

Several papers describe attention as a map from queries to context vectors, with exposure to keys and values that represent the input sequence [Vaswani *et al.*, 2017; Galassi *et al.*, 2019]. This model of attention makes no assumption about the type of input to the queries, keys, or values. The keys and queries are combined to create the attention vector, which is applied to the values to produce the context vector, the output of the attention mechanism. The keys and values can be seen as different representations of the same input [Galassi *et al.*, 2019], while the queries correspond to the current position of interest.

This model is so generic that little can be determined about how attention benefits models in practice. More experimentation will help us understand whether the varied manifestations of attention provide a common benefit, or if they are better considered separately.

3 Application

Here we review several recent applications of attention, hoping to observe common strengths and deficiencies. We also discuss directions that future work may take, given the strength that attention has shown in producing relationship metrics.

3.1 Attention without recurrent layers

[Vaswani *et al.*, 2017]

3.2 Multi-headed attention

[Vaswani *et al.*, 2017]

3.3 Model interpretation

Given that attention mechanisms learn relationships between sequences, we wonder if a simpler mechanism could learn the relationship from many inputs to a single output. One of the drawbacks of neural networks is that they don't provide a way to create feature importance metrics. These metrics are useful for determining why a model makes the decisions it does. In many papers cited here, attention produces a relationship matrix that provides intuition about the model's reasoning. It's possible that that same matrix could be produced by a simpler attention mechanism focused on the features that contribute to a single output value.

4 Conclusion

Currently, what is known for certain is that attention is useful. What is not known is exactly what it provides to the model. Is it simply a good way to generate a context vector, or does it successfully encode all the information needed to produce

the output? Is it simply a limited version of something more powerful?

References

[Bahdanau *et al.*, 2015] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2015.

[Cho *et al.*, 2014] Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax,Semantics and Structure in Statistical Translation*, pages 103–111, Doha, Qatar, oct 2014. Association for Computational Linguistics.

[Cybenko, 1989] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314, Dec 1989.

[Galassi *et al.*, 2019] Andrea Galassi, Marco Lippi, and Paolo Torroni. Attention, please! A critical review of neural attention models in natural language processing. *CoRR*, abs/1902.02181, 2019.

[Graves *et al.*, 2014] Alex Graves, Greg Wayne, and Ivo Danihelka. Neural Turing machines. *CoRR*, abs/1410.5401, 2014.

[Guru99, 2019] Guru99. Back propagation neural network: Explained with simple example. https://www.guru99.com/backpropogation-neural-network.html, 2019.

[Hassabis *et al.*, 2017] Demis Hassabis, Dharshan Kumaran, Christopher Summerfield, and Matthew Botvinick. Neuroscience-inspired artificial intelligence. *Neuron*, 95:245–258, 2017.

[Hecht-Nielsen, 1992] Robert Hecht-Nielsen. *Theory of the backpropagation neural network*. Elsevier, 1992.

[Kaiser and Bengio, 2016] Lukasz Kaiser and Samy Bengio. Can active memory replace attention? *CoRR*, abs/1610.08613, 2016.

[Kim *et al.*, 2017] Yoon Kim, Carl Denton, Luong Hoang, and Alexander M. Rush. Structured attention networks. *CoRR*, abs/1702.00887, 2017.

[Larochelle and Hinton, 2010] Hugo Larochelle and Geoffrey Hinton. Learning to combine foveal glimpses with a third-order boltzmann machine. In *Proceedings of the 23rd International Conference on Neural Information Processing Systems - Volume 1*, NIPS'10, pages 1243–1251, USA, 2010. Curran Associates Inc.

[Lin *et al.*, 2017] Zhouhan Lin, Minwei Feng, Cícero Nogueira dos Santos, Bing Xiang, Bowen Zhou, and Yoshua Bengio. A structured self-attentive sentence embedding. *CoRR*, abs/1703.03130, 2017.

[Nair and Hinton, 2010] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

[Parikh *et al.*, 2016] Ankur P. Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. A decomposable attention model for natural language inference. *CoRR*, abs/1606.01933, 2016.

[Ruder, 2016] Sebastian Ruder. An overview of gradient descent optimization algorithms. *CoRR*, abs/1609.04747, 2016.

[Vaswani *et al.*, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 30:5998–6008, 2017.