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.

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.

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.

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 impor-

[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].

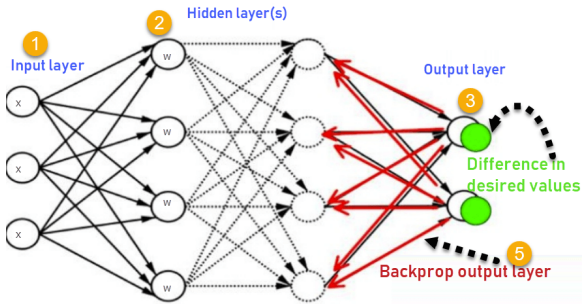

Figure 1: A visualization showing that backpropagation uses the difference between the desired and actual output to train the weights of the neural network. Image source [Guru99, 2019].

tant 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 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[3] 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"[4] 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[5], allowing the alignment to be performed by a feed-forward[6] neural network trained along with the rest of the model.

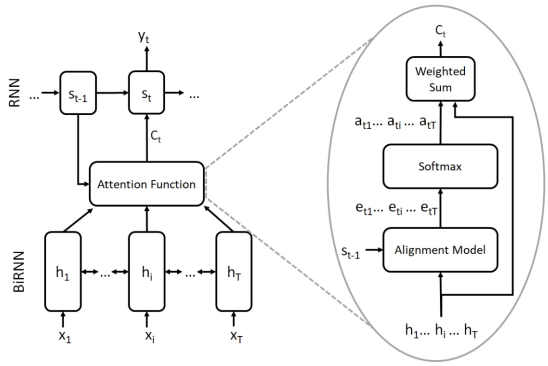

Figure 2: The model used by [Bahdanau *et al.*, 2015]. They used a bidirectional RNN to encode each word x_i into a fixed-length vector h_i. Then, for every position t in the target sentence, the alignment model produced an alignment vector a_t, a measure of alignment between the target location and 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 in terms of the objects it operates on, and second in terms of the information it provides.

2.1 Objects of attention

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 view attention beyond the scope of what the word "attention" implies.

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[7].

[3] i.e. a recurrent neural network, wherein the model is applied to each element of a sequence and also receives an encoding vector from the previous element.

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

[5] *differentiable* means that the function has a derivative, allowing gradients to pass through with backpropagation.

[6] *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, in which some output from a network is used as input to the same layer). An example representation can be seen in Figure 1.

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

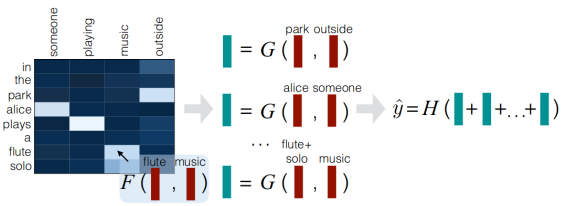

Figure 3: The model used by [Parikh *et al.*, 2016]. F is the attention model used to "decompose the problem into the comparison of aligned subphrases". G compares the aligned subphrases. Both F and G were feed-forward networks in the work.

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 what they term "intra-sentence attention" to represent relationships between words within the same 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, and provided for additional accuracy in the NLI model developed by Parikh *et al.* [2016].

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[8]. This allows the model to accept parse trees and other useful structures that explicitly encode relationships in the data.

This use of attention is relevant because it blurs the line between data and relationships. In the case of Kim *et al.* [2017], some known relationships were provided to the model in the form of parse tree structures. When the model was provided with this structure, the model learned to attend to entire subtrees as well as individual words. Thus the structure introduced in the data served as a way for the model to recognize and learn from information in the structural bias. Thus it was shown that attention mechanisms can learn from the inherent structure of their input.

More work should be given to understanding how attention learns from the structure of the objects it operates on. If attention can learn from the structure as well as the values of its input, any current application of attention could benefit from preprocessing the data to expose structural information. We've already seen this applied to natural language tasks [Kim *et al.*, 2017], but structure could be extracted from images, videos, and longer-form text input like paragraphs or web pages.

[8] 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, but in general a graph's elements can be connected to each other arbitrarily.

2.2 Information provided by attention

Discovering the hidden biases[9] of neural network models is an area of ongoing research in neural machine learning. Fortunately, much effort has been given to discovering the biases of attention models. Some limitations of attention have already been discovered, but others have only been hinted and deserve further work.

Active memory

One of the first applications of attention was to create a completely differentiable neural Turing machine[10] [Graves *et al.*, 2014]. The model was given memory to read from and write to at each iteration, 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 from exactly this issue, and offer a mechanism called active memory that attends to all inputs equally. They called their model the extended neural GPU, and it achieved promising results with better performance at translating long sentences. This could be due to the fact that long sentences are more likely to contain alignments that aren't one-to-one.

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 3.4).

Attention as a lookup mapping

Several papers describe attention as a map from a sequence of keys to a sequence of context vectors, with exposure to queries and values [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 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. See Figure 4 for more details on this model of attention. The greatest difference between this model and the original from Bahdanau *et al.* [2015] is the distinction between keys and values; Bahdanau *et al.*'s model uses the same representation h for both.

[9] A *bias* refers to the tendency of a network structure to exhibit a certain behavior, regardless of the weights the network. Understanding the biases of a network structure allows us to understand how it learns, and helps us predict strengths and weaknesses. See the last paragraph of Section 1.1 for a discussion of this aspect of neural network research.

[10] In computing theory, a *Turing machine* is any computer that can execute an arbitrary algorithm [Wikipedia, 2019].

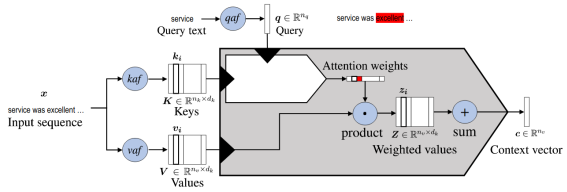

Figure 4: The generalized model of attention from [Galassi *et al.*, 2019]. Here kaf and vaf are functions mapping the input sequence to keys and values, respectively. An important distinction should be made between the terms "attention weights" and "context vector": attention weights are the "soft alignment" between the query and elements of the input sequence, and the context vector is the compiled information resulting from the weighted average of the values with respect to the attention weights. Image source [Galassi *et al.*, 2019].

This unified model generalizes the mechanism to include nearly all forms of attention found in literature. It describes commonalities between all variants of the mechanism, and offers the possibility of more completely describing the functionality. More effort is needed to describe some of the biases and uncover the basic functionality of attention in greater detail.

3 Application

Here we review how several recent applications of attention called into question previous assumptions about what attention accomplishes in a neural network model. We also discuss directions that future work may take, given the strength and generality that attention has shown in these works.

3.1 Dot-product attention

With the original form of attention developed by Bahdanau *et al.*[2015], the relationship between the jth input representation h_j and the ith output representation (query) s_i can be represented as

$$\text{attention}(h_j, s_i) = \langle v_j, a(W_1 h_j + W_2 s_i) \rangle \,,$$

where $\langle \cdot, \cdot \rangle$ is a dot product[11], v_j is the value representation, a is an activation function[12], and W_1, W_2 are trainable weight matrices [Britz *et al.*, 2017].

Luong and Manning [2016] introduced a modified mechanism that's faster and produces nearly the same performance. It is called *multiplicative attention*, in contrast with the original *additive attention* [Britz *et al.*, 2017]. Multiplicative attention is defined as

$$\text{attention}(h_j, s_i) = \langle W_1 h_j, W_2 s_i \rangle \,,$$

ignoring the value representation v_j. Additive attention "slightly but consistently outperformed" multiplicative attention in a battery of tests [Britz *et al.*, 2017], but the speed benefit of multiplicative attention is attractive enough that many later models used that implementation [Vaswani *et al.*, 2017; Sordoni *et al.*, 2016].

[11] i.e. a weighted sum.

[12] An *activation function* is a non-linear function typically used at the output of a neural network, so that it can learn non-linear mappings.

3.2 Attention without weights

In their Transformer model, Vaswani *et al.* [2017] further simplified the computation of attention by removing the learned weights W_1 and W_2. The queries, keys, and values are packed into matrices $Q = [q_1 \dots q_n]$, $K = [k_1 \dots k_m]$, and $V = [v_1 \dots v_m]$, and the mechanism is then

$$\text{attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right) V,$$

where d_k is the dimension of the query and key vectors.

Removing the weights from the mechanism means this component is not trainable at all. Surprisingly, it appears that the trainable parts of the Transformer are still able to learn from the output of the attention mechanisms, producing state-of-the-art results on machine translation [Vaswani *et al.*, 2017]. This implies that the value derived from attention is not in what it learns to "attend to" but in the information flow it provides to the rest of the network. In addition, Britz *et al.* [2017] have observed that attention models achieve "significantly larger gradient updates to decoder states throughout training", which supports the idea that attention should be considered primarily a mechanism for improving gradient flow.

This view of attention casts doubt upon the goal attention appeared to have in the past. If we no longer train the component to recognize relationships between inputs, how then does it contribute to the learning of the model? It's understood that attention improves gradient flow [Britz *et al.*, 2017], but research has yet to determine why it does so even without trainable parameters[13].

3.3 Attention without recurrent layers

RNNs are one of the most common models to use attention, and are especially slow to train due to their sequential nature. Vaswani *et al.* [2017] were able to avoid using RNNs entirely and created an encoder-decoder scheme using only weight-free attention (see Section 3.2) and linear layers. This model significantly improved the state of the art for machine translation, and in a fraction of the training time.

This means that attention does not necessarily rely on the sequential encoding characterized by RNNs. In fact, the only explicitly sequential information provided to the Transformer were the values of sinusoidal functions added to the input as "positional encodings". The model still had to learn the relationship between that encoding and the data itself. In the Transformer, attention mechanisms at each layer were able to provide the linear layers with access to the relationship information in a representation that they could easily learn from.

3.4 Multi-headed attention

One other innovation from Vaswani *et al.* [2017] was *multi-headed attention* wherein parallel layers of attention functions were each given a "different, learned linear projection"[14] of the same input data. If the input data is of dimen-

[13] i.e. weights.

[14] A *linear projection* is a linear function that maps high-dimensional input to a lower-dimensional space. A *learned linear projection* is a linear projection with trainable weights, allowing backpropagation to iteratively improve the projection.

sion d, an n-headed attention function with d/n-dimensional projections has a similar computational complexity to single-headed attention, but is able to simultaneously attend to multiple locations. This overcomes one of the inherent deficiencies of attention without decreasing speed.

3.5 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.

[Britz *et al.*, 2017] Denny Britz, Anna Goldie, Minh-Thang Luong, and Quoc 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, September 2017. Association for Computational Linguistics.

[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.

[Luong and Manning, 2016] Minh-Thang Luong and Christopher 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, August 2016. Association for Computational Linguistics.

[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.

[Sordoni *et al.*, 2016] Alessandro Sordoni, Philip Bachman, and Yoshua Bengio. Iterative alternating neural attention for machine reading. *CoRR*, abs/1606.02245, 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.

[Wikipedia, 2019] Wikipedia. Turing machine — Wikipedia, the free encyclopedia, 2019. [Online; accessed 13-Dec-2019].