# The Attention Mechanism – Review

D. Gueorguiev 1/7/2023

Table of Contents

[The Attention Mechanism – Review 1](#_Toc161722111)

[Introductory Notes 1](#_Toc161722112)

[Appendix 2](#_Toc161722113)

[Probability Density Estimation 2](#_Toc161722114)

[Bi-directional Neural Networks 2](#_Toc161722115)

[Deriving Machine Attention from Human Rationales 3](#_Toc161722116)

[Bibliography 3](#_Toc161722117)

## Introductory Notes

In many problems that involve the processing of natural language, the elements composing the source text are characterized by having each a different relevance to the task at hand. For instance, in aspect-based sentiment analysis, cue words, such as “good” or “bad”, could be relevant to some aspects under consideration but not to others. In machine translation, some words in the source text could be irrelevant in the translation of the next word. In a visual question-answering task, background pixels could be irrelevant in answering a question regarding an object in the foreground but relevant to questions regarding the scenery.

The effective solutions to such problems should factor in a notion of relevance, so as to focus the computational resources on a restricted set of important elements. One possible approach would be to tailor solutions to the specific genre at hand, on order to better exploit known regularities of the input, by feature engineering.

For example, in the argumentative analysis of persuasive essays, one could decide to give special emphasis to the final sentence. However, such an approach is not always viable, especially if the input is long or very information-rich, such as in text summarization, where the output is the condensed version of a possibly lengthy text sentence.

Another approach of increasing popularity amounts to machine learning the relevance of input elements. In that way, neural architectures could automatically weigh the relevance of any region of the input and take such a weight into account while performing the main task. The most common solution to this problem is a mechanism known as attention.

Attention was first introduced in NLP for machine translation tasks in [[4]](https://github.com/dimitarpg13/transformers_intro/blob/main/articles_and_books/NeuralMachineTranslationByJointlyLearningToAlignAndTranslateBahdanau2015.pdf). However, the idea of glimpses had already been proposed in Computer Vision by Larochelle and Hinton in [[19]](https://github.com/dimitarpg13/transformers_intro/blob/main/articles_and_books/NIPS-2010-learning-to-combine-foveal-glimpses-with-a-third-order-boltzmann-machine-Paper.pdf), following the observation that biological retinas fixate on relevant parts of the optic array, while resolution falls off rapidly with eccentricity. The term visual attention became especially popular after Mnih et al [[20]](https://github.com/dimitarpg13/transformers_intro/blob/main/articles_and_books/Recurrent_Models_of_Visual_Attention_Mnih_2014.pdf) significantly outperformed the state of the art in several image classification tasks as well as in dynamic visual control problems such as object tracking due to an architecture that could adaptively select and then process a sequence of regions or locations at high resolution and use a progressively lower resolution for further pixels.

Besides offering a performance gain, the attention mechanism can also be used as a tool for interpreting the behavior of neural architectures, which are notoriously difficult to understand. Indeed, neural networks are subsymbolic architectures; therefore, the knowledge they gather is stored in numeric elements that do not provide any means of interpretation by themselves. It is hard if not impossible to pinpoint the reasons behind the wrong output of a neural architecture. Interestingly, attention could be used to partially interpret and explain neural network behavior even if it cannot be considered a reliable means of explanation. For instance, the weights computed by attention could point us to relevant information discarded by the neural network or to irrelevant elements of the input source that have been factored in and could explain a surprising output of the neural network. Therefore, visual highlights of attention weights could be instrumental in analyzing the outcome of neural networks.

A text on a white background

Description automatically generated

Figure: Example of attention visualization for an aspect-based sentiment analysis task from Fig 6, [[21]](https://github.com/dimitarpg13/transformers_intro/blob/main/articles_and_books/Deriving_Machine_Attention_from_Human_Rationales_Bao_IBM-Watson_2018.pdf).

The attention mechanism is a part of a neural architecture that enables dynamically to select relevant features in the input data which in NLP is typically a sequence of textual elements. The idea behind attention is to compute a weight distribution on the input sequence, assigning higher values to more relevant elements. To illustrate we will look into classic attention architecture RNNsearch ([4]).

//TODO: finish this paragraph which corresponds to Section II of [1]

## Appendix

### Probability Density Estimation

We consider the problem of modelling a probability density function , given a finite number of data points drawn from that density function. The methods for probability density estimation are used to build classifier systems by considering each of the classes in turn and estimating the corresponding class-conditional densities by making use of the fact that each data point is labelled according to its class. These densities can then be used with Bayes theorem to find the posterior probabilities corresponding to a new measurement of , which can in turn be used to make a classification of **.**

We consider three alternative approaches to density estimation. The first of these involves *parametric* methods in which a specific functional form for the density model is assumed. This contains a number of parameters which are then optimized by fitting the model to the data set.

#### Parametric Methods

//TODO: finish this paragraph and the corresponding Appendix section which corresponds to Chapter 2 of [3]

### Notes on Neural Machine Translation

Unlike the traditional phrase-based translation system which consists of many small sub-components that are tuned separately, neural machine translation attempts to build and train a single, large network that reads a sentence and outputs a correct translation.

Most of the proposed neural machine translation models belong to a family of encoder-decoders, with an encoder and a decoder for each language, or involve a language-specific encoder applied to each sentence whose outputs are then compared. An encoder neural network reads and encodes a source sentence into a fixed-length vector.

A decoder then outputs a translation from the encoded vector. The whole encoder-decoder system, which contains of the encoder and the decoder for a language pair, is jointly trained to maximize the probability of a correct translation given a source sentence.

A potential issue with the encoder-decoder approach is that a neural network needs to be able to compress all he necessary information of a source sentence into a fixed-length vector. This may make it difficult for the neural network to cope with long sentences, especially those that are longer than the sentences in the training corpus. It was shown that indeed the performance of the traditional encoder-decoder architectures deteriorates with the length of an input sentence increases. In order to address this deterioration it has been introduced an extension mechanism to the traditional encoder-decoder model which learns to align and translate jointly. Each time the proposed model generates a word in a translation, it soft-searches for a set of positions in a source sentence where the most relevant information is concentrated. The model then predicts a target word based on the context vectors associated with these source positions and all the previous generated target words.

The most important distinguishing feature of this approach from the basic encoder-decoder is that it does not attempt to encode a while input sentence into a single fixed-length vector. Instead it encodes the input sentence into a sequence of vectors and chooses a subset of these vectors adaptively while decoding the translation. This frees a neural translation model from having to squash all the information of a source sentence, regardless of its length, into a fixed-length vector. Instead, it encodes the input sentence into a sequence of vectors and chooses a subset of these vectors adaptively while decoding the translation. This frees a neural translation model from having to squash all the information of a source sentence, regardless of its length, into a fixed-length vector. This allows the model to cope better with long sentences. This approach to jointly learning to align and translate achieves significantly improved translation performance over the basic encoder-decoder approach. The improvement is more apparent with longer sentences, but can be observed with sentences of any length.

#### Background

From a probabilistic perspective, translation is equivalent to finding a target sentence that maximizes the conditional probability of given a source sentence , i.e. . In neural machine translation a parametrized model is fit to maximize the conditional probability of sentence pairs using a parallel training corpus. Once the conditional distribution is learned by a translation model, given a source sentence a corresponding translation can be generated by searching for the sentence that maximizes the conditional probability.

Obviously neural networks can be used to directly learn this conditional distribution (see [7], [12], [13]).

The neural network approach consists of two components , the first of which encodes a source sentence , and the second decodes to a target sentence . For example, two recurrent neural networks (RNN) were used in [7] and [12] to encode a variable-length source sentence into a fixed-length vector and to decode the vector into a variable-length target sentence. In [12] long short term memory (LSTM) units were used to improve the performance of the translation. Neural components has been added to existing translation systems , for instance to score the phrase pairs in the phrase table ([7]) or to re-rank candidate translations ([12]) improves further the performance.

**RNN Encoder-Decoder**

This is the framework which was adopted by [7] and [12] which can be used as a base framework to add attention mechanism to it.

In the Encoder-Decoder framework, an encoder reads the input sentence, a sequence of vectors into a vector . The most common approach is to use an RNN such that

(nmt.1)

and

,

where is a hidden state at time , and is a vector generated from the sequence of the hidden states. Sutskever et al (2014) ([[22]](https://github.com/dimitarpg13/transformers_intro/blob/main/articles_and_books/SequencetoSequenceLearningwithNeuralNetworksSutsekver2014.pdf)) used an LSTM as and for instance.

The decoder is often trained to predict the next word given the context vector and all the previously predicted words . In other words, the decoder defines a probability over the translation by decomposing the joint probability into the ordered conditionals:

(nmt.2)

where . With an RNN, each conditional probability is modeled as:

(nmt.3)

where is a non-linear, potentially multi-layered, function (that is multilayered NN) that outputs the probability of , and is the hidden state of the RNN.

#### Learning to Align and Translate

The new neural architecture proposed by Bahdanau et al in [4] consists of bidirectional RNN as an encoder and a decoder that emulates searching through a source sentence during decoding a translation.

//TODO: finish the section on Neural Machine Translation (mainly from [4])

### Bi-directional Neural Networks

*Problem Statement*

Consider a (time) sequence of input data vectors

and a sequence of corresponding output data vectors

with neighboring data-pairs in time being statistically independent. Given time sequences and as training data, the aim is to learn the rules to predict output data given the input data. Inputs and outputs, can, in general, be continuous and/or categorical variables. When the outputs are continuous, we have *regression problem* at hand and when they are categorical (class labels), the problem is known as a *classification problem*. In general we talk about *prediction problem* which includes regression and classification.

1. *Unimodal Regression*

With unimodal regression or function approximation, the components of the output vectors are continuous variables. The network parameters are estimated to maximize some predefined objective criterion e.g. maximize the likelihood of the output data. When the distribution of the errors between the desired and the estimated output vectors is assumed to be Gaussian with zero mean and a fixed global data-dependent variance, the likelihood criterion reduces to the Euclidean distance measure between the desired and the estimated output vectors or the *mean-squared-error criterion*, which has to be minimized during training.

Figure brnn.1: Structure of regular unidirectional RNN shown (a) with a delay line and (b) unfolded in time with two time steps

output neuron

group

hidden state

neuron group

inputs

delay line

group of weights

with information flow

It has been shown that neural networks can estimate the conditional average of the desired output (or target) vectors at their network outputs i.e. .

1. *Classification*

In the case of a classification problem, one seeks the most probable class out of a given pool of classes for every time frame , given an input vector sequence . To make this kind of problem suitable to be solved by an NN, the categorical variables are usually coded as vectors as follows. Consider that is the desired class label for the frame at time . Then, construct an output vector such that its th component is one and other components are zero.

The output vector sequence constructed in this manner along with the input vector sequence can be used to train the network under some optimality criterion, usually cross-entropy criterion, which results from a maximum likelihood estimation assuming a multinomial output distribution.

It has been shown that the th network output at each time point can be interpreted as an estimate of the conditional posterior probability of class membership for class , with the quality of the estimate depending on the size of the training data and he complexity of the network.

For some applications, it is not necessary to estimate the conditional posterior probability of a *single class* given the sequence of input vectors but the conditional posterior probability of a *sequence* of classes given the sequence of input vectors.

Given a series of paired input/output vectors we want to train bidirectional recurrent neural nets (BRNN) to perform the following tasks:

1 ) Unimodal regression – compute

2 ) Classification – compute for every output class and decide the class using the maximum *a posteriori* decision rule. In this case the outputs are treated as statistically independent.

3 ) Estimation of the conditional probability of a complete sequence of classes of length T using all available input information i.e. compute . In this case, the outputs are treated as being statistically dependent, which makes the estimation more difficult and requires slightly different network structure.

#### Prediction Assuming Independent Outputs

*Recurrent Neural Networks (RNNs)*

RNNs provide a way of dealing with time sequential data that embodies correlations between data points that are close in the sequence. Figure brnn.1 shows a basic RNN architecture with a delay line and unfolded in time for two time steps. In this structure, the input vectors are fed one at a time into the RNN. Instead of using a fixed number of input vectors as done in MLP and TDNN, this architecture can make use of all the available input information up to the current time frame (i.e. ) to predict . How much of this information is captured by a particular RNN depends on its structure and the training algorithm.

The amount of input information used for prediction with different kinds of NN’s is given as

MLP/TDNN: ,

RNN:

Forward RNN with delay:

Backward RNN with delay: , , – the index of the final frame

BRNN (Bidirectional RNN): , – the index of the final frame

Future input information coming up later than is usually also useful for prediction. With an RNN, this can be partially achieved by delaying the output by a certain number of M time frames to include future information up to

to predict . could be made very large to capture all the available future information, but in practice, it is found that prediction results drop if is too large. A possible explanation for this could be that with rising , the modeling power of the RNN is increasingly concentrated on remembering the input information up to for the prediction of , leaving less modeling power for combining the prediction knowledge from different input vectors. The optimal delay is task dependent and has to be found by the trial and error method on the validation test set. More elegant approach is desirable.

To use all available input information, it is possible to use two separate networks (one for each time direction) and then somehow merge the results. Both networks can then be called experts for the specific problem on which the networks are trained. Both networks can then be called *experts* for the specific problem on which the networks are trained. One way of *merging the opinions of different experts* is to assume the opinions to be independent, which leads to arithmetic averaging for regression and to geometric averaging (or equivalently arithmetic averaging in log domain) for classification.

output neuron

group

hidden state

neuron group

inputs

group of weights

with information flow

forward

states

backward

states

//TODO: finish this paragraph and the corresponding Appendix section which corresponds to Section B of [2]

### Deriving Machine Attention from Human Rationales

Attention-based models are successful when trained on large amounts of data. Attention is often used as a proxy for human interpretable rationales.

In [[21]](https://github.com/dimitarpg13/transformers_intro/blob/main/articles_and_books/Deriving_Machine_Attention_from_Human_Rationales_Bao_IBM-Watson_2018.pdf) it is demonstrated that even with small amounts of data attention can be learned effectively. The authors start with discrete human annotated rationales and map those into continuous attention. The premise is that the mapping is general across domains, and this can be transferred from resource-rich domains to low-resource ones. The model presented in [21] jointly learns a domain-invariant representation and induces the desired mapping between rationales and attention.

The notion of rationale and attention are closely related. Both of them highlight word importance for the final prediction. In the case of rationale, the importance is expressed as a hard selection, while attention provides a soft distribution over the words. The Figure below illustrates this relatedness:

A close up of a text

Description automatically generatedFigure: Examples of rationales versus oracle attention. Words are highlighted according to their relative attention scores. Human rationales are shown in bold with underlines.

One obvious approach to improve low-resource performance (that is, small amounts of training data) is to directly use human rationales as a supervision for attention generation. The implicit assumption behind this method is that machine-generated attention should mimic human rationales. However, rationales on their own are not adequate substitutes for machine attention. Instead of providing soft distribution human rationales only provide the binary indication about relevance. Furthermore, rationales are subjectively defined and often vary across annotators. Finally, human rationales are not customized for a given model architecture. In contrast, machine attention is always derived as a part of a specific model architecture.

To further understand the connections, the authors of [[21]](https://github.com/dimitarpg13/transformers_intro/blob/main/articles_and_books/Deriving_Machine_Attention_from_Human_Rationales_Bao_IBM-Watson_2018.pdf) empirically compare models informed by human rationales and those by high-quality attention. To obtain high-quality attention an *oracle attention* concept is defined by the authors who used large amounts of annotations for it. This *oracle attention* is then

//TODO: finish this paragraph and the corresponding Appendix section which corresponds to Section x of [21]

## Bibliography

[1] [Attention in Natural Language Processing, Andrea Galassi, Marco Lippi, Paolo Torroni, 2019](https://github.com/dimitarpg13/transformers_intro/blob/main/articles_and_books/AttentionInNaturalLanguageProcessing.pdf)

[2] [Bidirectional Recurrent Neural Networks, Mike Schuster, Kuldip Paliwal, IEEE Transactions on Signal Processing, 1997](https://github.com/dimitarpg13/transformers_intro/blob/main/articles_and_books/Bidirectional_Recurrent_Neural_Networks_Schuster_Paliwal_1997.pdf)

[3] [Neural Networks for Pattern Recognition, Christopher M. Bishop, 1995](https://github.com/dimitarpg13/transformers_intro/blob/main/articles_and_books/Neural_Networks_for_Pattern_Recognition-Bishop_1995.pdf)

[4] [Neural Machine Translation by Jointly Learning To Align and Translate, Dzmitry Bahdanau, K. Cho, Yoshua Bengio, 2016](https://github.com/dimitarpg13/transformers_intro/blob/main/articles_and_books/NeuralMachineTranslationByJointlyLearningToAlignAndTranslateBahdanau2015.pdf)

[5] [Recurrent Continuous Translation Models, Nal Kalchbrenner, Phil Blunsom, Oxford U., 2013](https://github.com/dimitarpg13/transformers_intro/blob/main/articles_and_books/Recurrent_continuous_translation_models_Kalchbrenner_Blunsson_OxfordU_2013.pdf)

[6] [On The Properties of Neural Machine Translation: Encoder-Decoder Approaches, K. Cho, B. van Merrienboer, Dzmitry Bahdanau, 2014](https://github.com/dimitarpg13/transformers_intro/blob/main/articles_and_books/OnthePropertiesOfNeuralMachineTranslationEncoderDecoderApproaches.pdf)

[7] [Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation, K. Cho, B. van Merrienboer, C. Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, Yoshua Bengio, 2014](https://github.com/dimitarpg13/transformers_intro/blob/main/articles_and_books/LearningPhraseRepresentationsUsingRNNEncoderDecoderForStatisticalMachineTranslation.pdf)

[8] [Autoencoders, Dor Bank, Noam Koenigstein, Raja Giryes, 2021](https://github.com/dimitarpg13/transformers_intro/blob/main/articles_and_books/Autoencoders.pdf)

[9] [Attention and Augmented Recurrent Neural Networks, Chris Olah, Shan Carter, Google Brain blog, 2016](https://github.com/dimitarpg13/transformers_intro/blob/main/articles_and_books/AttentionandAugmentedRecurrentNeuralNetworks.pdf)

[10] [Attention is All You Need, Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit et al, NIPS, 2017](https://github.com/dimitarpg13/transformers_intro/blob/main/articles_and_books/Attention-is-all-you-need-NIPS-2017.pdf)

[11] [Transforming Auto-encoders, G.E. Hinton, A. Krizhevsky, S.D. Wang, U of Toronto, 2010](https://github.com/dimitarpg13/transformers_intro/blob/main/articles_and_books/TransformingAutoencodersHinton.pdf)

[12] [Sequence to Sequence Learning with Neural Networks, Ilya Sutskever, Oriol Vinyals, Quoc V. Le, Google Research, 2014](https://github.com/dimitarpg13/transformers_intro/blob/main/articles_and_books/SequencetoSequenceLearningwithNeuralNetworksSutsekver2014.pdf)

[13] [Generating Sequences With Recurrent Neural Networks, Alex Graves, U of Toronto, 2014](https://github.com/dimitarpg13/transformers_intro/blob/main/articles_and_books/Generating_Sequences_With_Recurrent_Neural_Networks_Graves_2014.pdf)

[14] [A Tutorial on Training Recurrent Neural Networks, covering BPPT, RTRL, EKF and the “echo state network” approach, Herbert Jaeger, Fraunhofer Institute for Autonomous Intelligent Systems, 2003](https://github.com/dimitarpg13/transformers_intro/blob/main/articles_and_books/TutorialOnRNNAndBPTTJaeger2002.pdf)

[15] [Understanding LSTM: a Tutorial into Long Short-Term Memory Recurrent Neural Networks, Ralf C. Staudemeyer, Eric Rothstein Morris, 2019](https://github.com/dimitarpg13/transformers_intro/blob/main/articles_and_books/TutorialOnLongShortTermMemory2019.pdf)

[16] [Long Short-Term Memory, Sepp Hochreiter, Juergen Schmidhuber, Neural Computation 9(8), 1997](https://github.com/dimitarpg13/transformers_intro/blob/main/articles_and_books/LongShortTermMemory.pdf)

[17] [Sequence Modeling With Neural Networks (Part 1): Language & Seq2Seq, Nathan Lintz’s blog, 2016](https://github.com/dimitarpg13/transformers_intro/blob/main/articles_and_books/SequenceModelingWithNeuralNetworksPart1_Seq2Seq.pdf)

[18] [Sequence Modeling With Neural Networks (Part 2): Attention Models, Nathan Lintz’s blog, 2016](https://github.com/dimitarpg13/transformers_intro/blob/main/articles_and_books/SequenceModelingwithNeuralNetworksPart2_AttentionModels.pdf)

[19] [Learning to combine foveal glimpses with a third-order Boltzmann machine, H. Larochelle and G. Hinton, 2010](https://github.com/dimitarpg13/transformers_intro/blob/main/articles_and_books/NIPS-2010-learning-to-combine-foveal-glimpses-with-a-third-order-boltzmann-machine-Paper.pdf)

[20] [Recurrent models of visual attention, V. Mnih et al, Google DeepMind, 2014](https://github.com/dimitarpg13/transformers_intro/blob/main/articles_and_books/Recurrent_Models_of_Visual_Attention_Mnih_2014.pdf)

[21] [Deriving Machine Attention from Human Rationales, Y. Bao et al, CSAIL-MIT, IBM Watson, 2018](https://github.com/dimitarpg13/transformers_intro/blob/main/articles_and_books/Deriving_Machine_Attention_from_Human_Rationales_Bao_IBM-Watson_2018.pdf)

[22] [Sequence to Sequence Learning with Neural Networks, Sutskever et al, Google Research, 2014](https://github.com/dimitarpg13/transformers_intro/blob/main/articles_and_books/SequencetoSequenceLearningwithNeuralNetworksSutsekver2014.pdf)