attention

Authors

Alexander Jung*, Department of Computer Science, Aalto University, Espoo, Finland
*Corresponding author: alex.jung@aalto.fi

Synonyms

attention mechanism, attention head

Abstract

Attention is a mechanism that models the dependencies between the tokens that make up a data point, such as the words in a sentence or the pixel patches of an image. The idea is to represent the relationship between two tokens by a parameterized weight function whose model parameters are learned from a training set. An attention head then computes a new vector representation for each token as a weighted combination of the value vectors of all tokens. Like the weight function, the value vectors are learned. An attention head acts as a differentiable associative memory: a token uses its query vector to retrieve, through the key vectors of the other tokens, the values most relevant to it. Attention heads capture long-range dependencies between tokens regardless of their positions within a data point, and they are a core component of modern large language models (LLMs).

Definition

Some machine learning (ML) applications involve data points composed of smaller units, referred to as tokens. For example, a sentence consists of words, an image of pixel patches, and a network of nodes. In general, the tokens that constitute a single data point are not independent of one another. Instead, each token of a data point depends on specific other tokens of the same data point. The attention mechanism is a building block of artificial neural networks (ANNs) that captures long-range dependencies between tokens regardless of their positions within a data point.

Probabilistic models provide a principled way of representing and analyzing such dependencies (Blei et al., 2003). Attention mechanisms instead represent these dependencies directly, without specifying a probability distribution over the tokens. They represent the relationship between two tokens i and i' by an attention weight $f^{(\mathbf{w})}(i, i')$, a parameterized function whose model parameters $\mathbf{w}$ are learned. The attention weight measures how strongly token i attends to token i'. Practical attention mechanisms differ in their choice of the function $f^{(\mathbf{w})}(i, i')$ and in the empirical risk minimization (ERM) variant used to learn $\mathbf{w}$. The most widely used choice, described below, is scaled dot-product attention (Vaswani et al., 2017).

Scaled dot-product attention derives three vectors from the embedding vector $\mathbf{x}^{(i)} \in \mathbb{R}^d$ of each token i via learned projection matrices: a query (vector) $\mathbf{q}^{(i)} = \mathbf{W}_Q \mathbf{x}^{(i)}$, a key (vector) $\mathbf{k}^{(i)} = \mathbf{W}_K \mathbf{x}^{(i)}$, and a value (vector) $\mathbf{v}^{(i)} = \mathbf{W}_V \mathbf{x}^{(i)}$, where

$$\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V \in \mathbb{R}^{d \times d}$$

are model parameters learned during training. Intuitively, the query determines what a token "asks for," the key of a token determines what it "advertises," and the value carries the information that is aggregated (cf. Bishop and Bishop, 2024; Elhage et al., 2021).

The output of the attention mechanism for token i is a weighted sum of the values of all tokens,

$$\mathbf{z}^{(i)} = \sum_{i'=1}^{n} \alpha_{i,i'} \, \mathbf{v}^{(i')}.$$

Here, n denotes the number of tokens of the data point, and the coefficient $\alpha_{i,i'}$ is the attention weight, i.e., the scaled dot-product form of $f^{(\mathbf{w})}(i, i')$. The attention weights are computed from the queries and keys in two steps. First, the attention score between tokens i and i' is

$$s_{i,i'} = \frac{\left(\mathbf{q}^{(i)}\right)^\top \mathbf{k}^{(i')}}{\sqrt{d}}.$$

The inner product (also called dot product) $\left(\mathbf{q}^{(i)}\right)^\top \mathbf{k}^{(i')}$ measures how well the key of token i' matches the query of token i. Second, the attention weight $\alpha_{i,i'}$ follows by applying the softmax function to the scores $s_{i,i'}$ across all tokens i':

$$\alpha_{i,i'} = \frac{\exp(s_{i,i'})}{\sum_{j=1}^{n} \exp(s_{i,j})}.$$

The softmax function normalizes the scores: the attention weights $\alpha_{i,i'}$ are nonnegative and sum to one across $i' = 1, \ldots, n$, forming a probability distribution over the tokens. Dividing the inner product by $\sqrt{d}$ in the attention score counteracts its growth with the dimension d (Vaswani et al., 2017, Sect. 3.2.1). Indeed, $\left(\mathbf{q}^{(i)}\right)^\top \mathbf{k}^{(i')}$ is a sum of d terms, so without the scaling, a large d would push the softmax function into a region where its gradients nearly vanish. Fig. 1 illustrates scaled dot-product attention for an eight-token sentence.

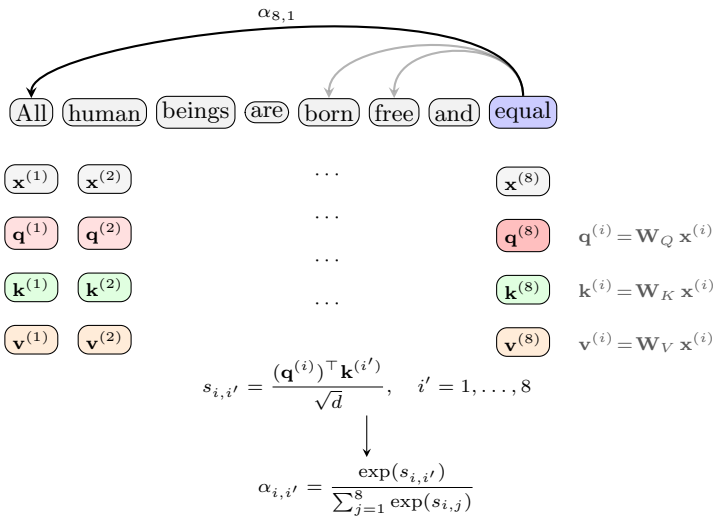

Figure 1: Scaled dot-product attention for the sentence "All human beings are born free and equal." Each token embedding $\mathbf{x}^{(i)}$ is projected by learned matrices $\mathbf{W}_Q$, $\mathbf{W}_K$, $\mathbf{W}_V$ into query, key, and value vectors. First, the attention score $s_{i,i'}$ is computed as the scaled dot product of query i and key i'. Then, the softmax function normalizes these scores into attention weights $\alpha_{i,i'}$ that sum to one across all positions. Arcs at the top illustrate how token 8 ("equal") attends to earlier tokens; thicker arcs indicate higher attention weights (Vaswani et al., 2017). The token "equal" and its query $\mathbf{q}^{(8)}$ are highlighted by a darker fill: the depicted weights $\alpha_{8,i'}$ involve the query of token 8 but the keys and values of all tokens

Moving from a single token to the whole sequence of tokens, an attention head is a parameterized function $h^{(\mathbf{w})}$ that maps the input embeddings to the output vectors,

$$h^{(\mathbf{w})}\big(\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)}\big) = \big(\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(n)}\big),$$

with model parameters $\mathbf{w}$ comprising the three projection matrices $\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V$. Collecting the output vectors as the rows of a matrix $\mathbf{Z}$, and the queries, keys, and values as the rows of $\mathbf{Q}$, $\mathbf{K}$, $\mathbf{V}$, $h^{(\mathbf{w})}$ computes them in the compact form

$$\mathbf{Z} = \mathrm{softmax}\left(\frac{\mathbf{Q}\,\mathbf{K}^\top}{\sqrt{d}}\right)\mathbf{V}.$$

Here, the softmax function is applied independently to each row of $\mathbf{Q}\,\mathbf{K}^\top/\sqrt{d}$, so that each row of the resulting matrix is a probability distribution over the n tokens. Since the queries, keys, and values are all derived from the same sequence of tokens, this construction is called self-attention. In practice, several attention heads, each with its own projection matrices, operate in parallel; this

is called multi-head attention, and each head can specialize to a different relation between tokens (Vaswani et al., 2017). Computing all pairwise scores $s_{i,i'}$ requires n^2 query–key inner products, one for each ordered pair of tokens.

Attention can be read as a differentiable associative memory, i.e., a content-addressable store (Ramsauer et al., 2021): the keys $\mathbf{k}^{(i')}$ act as addresses, the values $\mathbf{v}^{(i')}$ as stored contents, and a token i searches the other tokens with its query $\mathbf{q}^{(i)}$. The inner product $\left(\mathbf{q}^{(i)}\right)^\top \mathbf{k}^{(i')}$ measures how well the address i' matches the query, the softmax function turns the match scores into retrieval weights, and the output $\mathbf{z}^{(i)}$ is the retrieved content.

Because $\mathbf{q}^{(i)}$ and $\mathbf{k}^{(i')}$ are obtained from different projection matrices, the attention weights are *directed*: in general, $\alpha_{i,i'} \neq \alpha_{i',i}$, so that token i attending strongly to token i' does not imply the reverse. Directionality arises because asking for information (the query) and advertising it (the key) are distinct roles.

A classical dictionary data structure or hash table returns the single value stored at an exactly matching address (Cormen et al., 2022, Ch. 11). Attention instead returns a weighted average. Moreover, its output $\mathbf{z}^{(i)}$ is a differentiable function of the projection matrices $\mathbf{W}_Q$, $\mathbf{W}_K$, and $\mathbf{W}_V$. In particular, the queries, keys, and values can themselves be learned via gradient-based methods for solving ERM. The same retrieval principle, softened into a weighted average, underlies k-nearest neighbors (k-NN) methods. It also guides efficient implementations: locality-sensitive hashing restricts each query to nearby keys, reducing the cost of one attention head from n^2 query–key comparisons to approximately $n \log n$, with n again the number of tokens (Kitaev et al., 2020).

Fig. 2 visualizes the attention weights $\alpha_{i,i'}$ that a single head learns on sentences from the Universal Declaration of Human Rights (United Nations General Assembly, 1948). Trained to reconstruct each masked token from the others, the head gives many query tokens a weight concentrated on a few keys rather than spread uniformly, illustrating the associative-memory reading on this small corpus. The heat map also shows the directed nature of the weights discussed above: the query token "rights" places its full weight on the key token "beings" ($\alpha_{12,3} = 1.0$), while the query token "beings" places no weight on the key token "rights" ($\alpha_{3,12} = 0.0$).

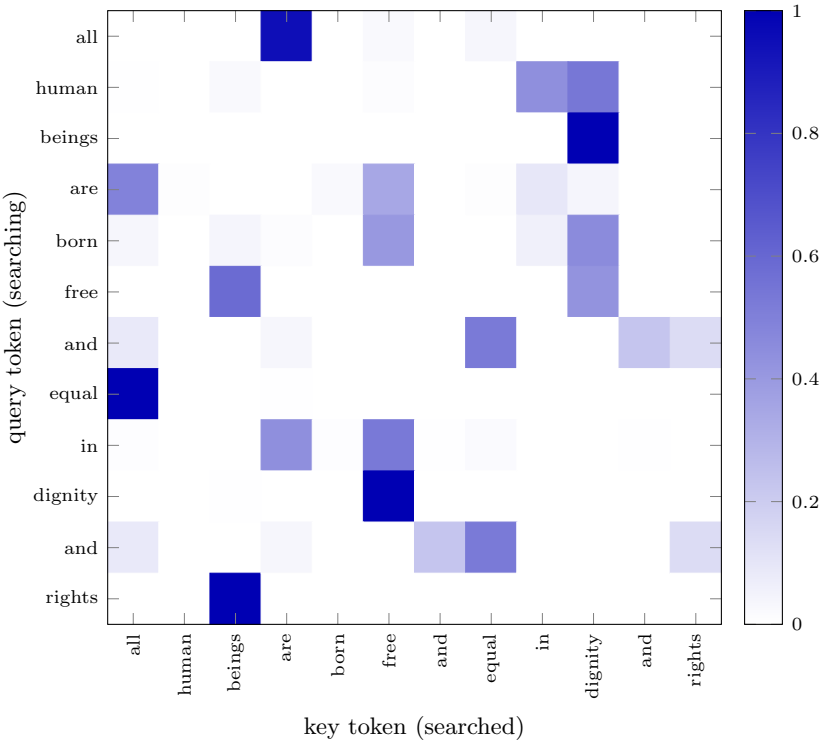

Figure 2: Attention weights $\alpha_{i,i'}$ of one self-attention head for the sentence "All human beings are born free and equal in dignity and rights" (United Nations General Assembly, 1948, Art. 1). Row i is the query token; column i' is the searched key token. Darker cells carry more weight: for instance, the query token "all" places nearly all of its weight on the key token "are" ($\alpha_{1,4} = 0.94$), and its remaining weights are close to zero, since the weights of each query sum to one. The head was trained on sentences from the Universal Declaration of Human Rights by masking each token and reconstructing it from the others, so the query must retrieve associated tokens through their keys. Data generated by `pythondemos/attention.py`.

Cross References

token, embedding, transformer, softmax function, sequence, NLP, LLM, model parameter, ERM, k-NN, differentiable

References

1. Bishop and Bishop (2024). Deep Learning: Foundations and Concepts. Springer Nature, Cham, Switzerland.

2. Blei et al. (2003). Latent Dirichlet Allocation. J. Mach. Learn. Res., 3, pp. 993–1022.

3. Cormen et al. (2022). Introduction to Algorithms. 4th ed., MIT Press, Cambridge, MA, USA.

4. Elhage et al. (2021). A Mathematical Framework for Transformer Circuits. Transformer Circuits Thread.

5. United Nations General Assembly (1948). Universal Declaration of Human Rights (UDHR). United Nations, New York.

6. Kitaev et al. (2020). Reformer: The Efficient Transformer. In: Int. Conf. Learn. Represent. (ICLR).

7. Ramsauer et al. (2021). Hopfield networks is all you need. In: Int. Conf. Learn. Represent. (ICLR).

8. Vaswani et al. (2017). Attention is all you need. In: Adv. Neural Inf. Process. Syst., pp. 5998–6008.