Transformers and Large Language Models

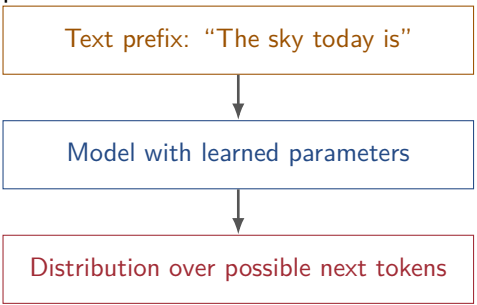
Mathematical foundations and model architecture

Contents

1. Introduction
2. Learning and parameters
3. Numbers and geometry
4. Tokenization and embeddings
5. Geometry of meaning
6. Skalarprodukt
7. Context
8. Transformer architecture
9. Unembedding
10. Softmax and temperature
11. References

GPT: three terms, one model family

Large Language Model (LLM): a large neural language model that learns statistical patterns in text.



G Generative
Produces new token sequences.

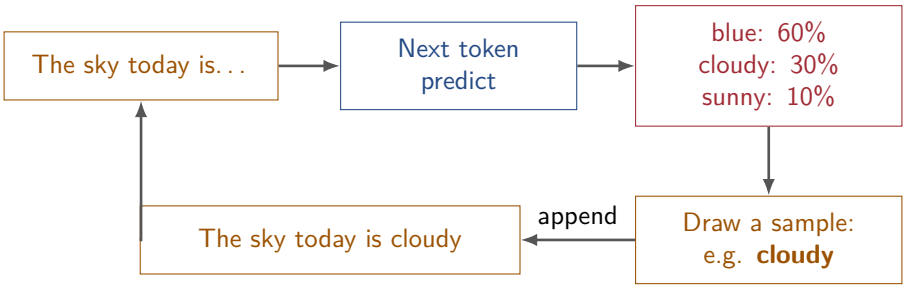
P Pretrained
Initially learns from many training examples.

T Transformer
A network architecture that processes representations in context.

Key idea: Here we study **autoregressive GPT language models**: They generate one token at a time.

[1] Not every LLM is a GPT. Chat assistants may undergo additional post-training.

Text generation through repeated prediction



Key idea: Sampling selects a token at random according to the probabilities. The second most likely token can also be selected.

[1] Constructed three-token vocabulary; real models assign probability to many more tokens.
Autoregressive: earlier tokens become input for the next step. Stop at an end token or a length limit.

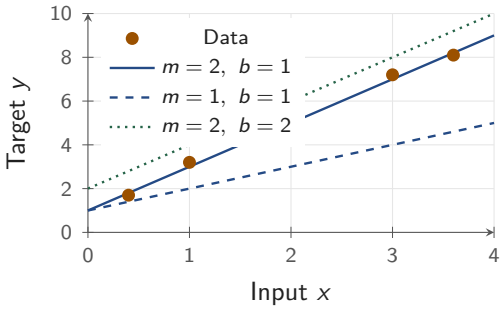
Machine learning starts with adjustable parameters

Machine Learning: Learn behaviour from data rather than specifying every rule.

$$\hat{y} = mx + b$$

Linear regression: fit a straight line to data.

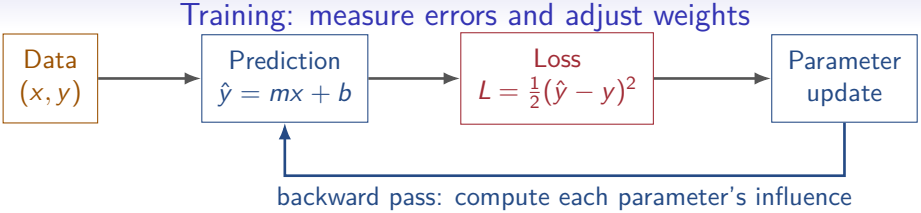
x: Input; $\hat{y}$: Prediction.
m: slope; b: intercept.



$$x = 2, \quad m = 2,\ b = 1$$
$$\Rightarrow \quad \hat{y} = 5$$

Key idea: A **parameter** is a learned adjustable value: changing it changes the predictions.

[1] Supplementary linear regression example. All data points are constructed.



Backpropagation computes derivatives using the chain rule.

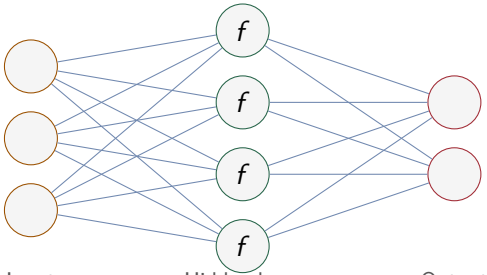
A **gradient** describes how the loss changes when parameters change.

One data point: $x = 2,\ y = 5$; Initial values: $m = 1, b = 0$.
$\hat{y} = 2$, error $\hat{y} - y = -3$, $L = 4.5$.

$$\frac{\partial L}{\partial m} = -3 \cdot 2 = -6,$$
$$\frac{\partial L}{\partial b} = -3$$

Gradient descent, learning rate $\eta = 0.1$:
$m' = 1 - 0.1(-6) = 1.6$, $b' = 0.3$.
New prediction: 3.5; new loss: $L' = 1.125$.

Neural networks combine simple computations



$$h_j = f\left(\sum_i w_{ji} x_i + b_j\right)$$

Weights scale contributions. A **Bias** shifts the sum.

An **activation function** f introduces a nonlinear transformation, e.g. $f(s) = \max(0, s)$.

Without nonlinearity, several affine layers would still compose to **one affine map**.

More parameters increase capacity, but do not guarantee quality. Data, architecture and training must work together.

Supplement: schematic network with ReLU. Overfitting means fitting training data while generalizing poorly to new data.

Scalars, vectors and matrices are tensors

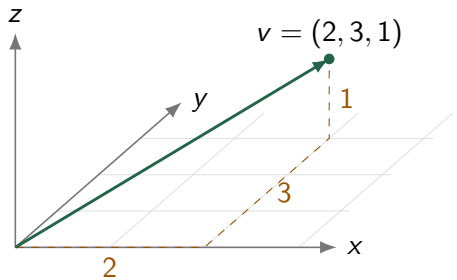
25	$\begin{pmatrix} 2 \\ 3 \\ 1 \end{pmatrix}$	$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$	A_{ijk}
Scalar one number Order 0	**Vector** one axis Order 1	**Matrix** two axes Order 2	**3D-Tensor** three axes Order 3

more indices are needed to select one entry

Key idea: In deep learning, a **tensor** is a numerical array. Its order counts the axes; its **shape** gives their lengths.

Supplement: tensor rank here means the number of axes, not matrix rank or algebraic tensor rank. A vector with 3 components has shape (3) and order 1.

A vector is a list, a point and an arrow



$$v = \begin{pmatrix} 2 \\ 3 \\ 1 \end{pmatrix} \in \mathbb{R}^3$$

Components: steps along the three axes.

Direction: from the origin to the point.

Length (Euclidean norm):

$$\|v\| = \sqrt{2^2 + 3^2 + 1^2}$$
$$= \sqrt{14} \approx 3.74.$$

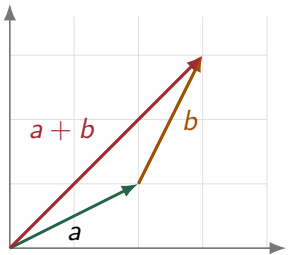
Key idea: The three numbers specify a position; the same vector can be drawn as an arrow from the origin.

Supplement: a three-dimensional example shown as a perspective drawing on a two-dimensional page.

Vector addition: join arrows head to tail

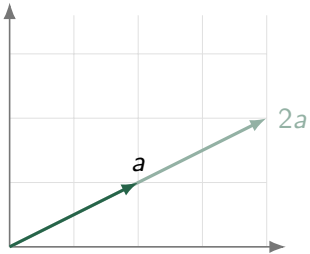
Addition

$$\begin{pmatrix}2\\1\end{pmatrix} + \begin{pmatrix}1\\2\end{pmatrix} = \begin{pmatrix}3\\3\end{pmatrix}$$



Scaling

$$2\begin{pmatrix}2\\1\end{pmatrix} = \begin{pmatrix}4\\2\end{pmatrix}$$



Key idea: Addition works component by component. A positive factor changes the length; a negative factor reverses the direction.

Supplement: 2D simplifies the drawing; the same rules apply in any dimension.

A matrix combines multiple weighted sums

$W \in \mathbb{R}^{3\times 3}$

$$\begin{bmatrix} 1 & 2 & 0 \\ 0 & 1 & 1 \\ 2 & 0 & 1 \end{bmatrix} \leftarrow \text{row 2}$$

3 rows, 3 columns
W_{ij}: row i, column j

Matrix-vector product: Each row produces one output component.

$$y_i = \sum_{j=1}^{3} W_{ij} x_j$$



An $m \times n$ matrix maps n input components to m output components.

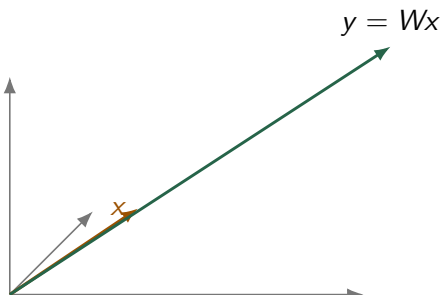
Key idea: The number of columns of W must match the length of x: $(m \times n)(n \times 1) = (m \times 1)$.

Supplement: throughout this presentation, vectors are column vectors.

A matrix product, step by step

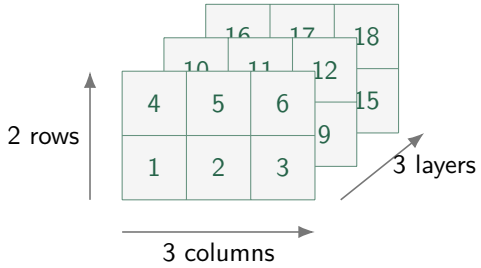
$$\underbrace{\begin{pmatrix} 8 \\ 4 \\ 5 \end{pmatrix}}_{\text{Output } y} = \underbrace{\begin{pmatrix} 1 & 2 & 0 \\ 0 & 1 & 1 \\ 2 & 0 & 1 \end{pmatrix}}_{\text{Weights } W} \underbrace{\begin{pmatrix} 2 \\ 3 \\ 1 \end{pmatrix}}_{\text{Input } x}$$

$$y_1 = 1 \cdot 2 + 2 \cdot 3 + 0 \cdot 1 = 8,$$
$$y_2 = 0 \cdot 2 + 1 \cdot 3 + 1 \cdot 1 = 4,$$
$$y_3 = 2 \cdot 2 + 0 \cdot 3 + 1 \cdot 1 = 5.$$



Key idea: The same calculation can be understood as **weighted sums** or as a **geometric**

A tensor adds further axes to a numerical array



Three indices select one entry:

$$A_{ijk}, \qquad \text{Shape } (2, 3, 3).$$

The block contains $2 \cdot 3 \cdot 3 = 18$ numbers.

In a language model:

$$\underbrace{B}_{\text{texts}} \times \underbrace{n}_{\text{Tokens}} \times \underbrace{d}_{\text{features}}$$

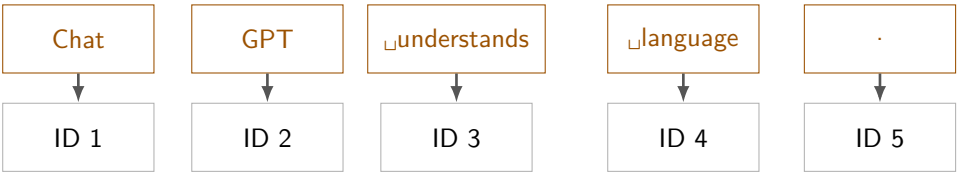
A **batch** groups B examples. Each text has n token vectors of length d.

Key idea: A tensor axis need not represent a spatial direction: it can index texts or token positions.

Supplement: below, we consider one text and arrange its token vectors as columns in $\mathbb{R}^{d \times n}$.

Tokenization: split text into addressable units

ChatGPT understands language.



A **token** can be a word, a word fragment, punctuation or another text unit.

The **vocabulary** is the fixed list of possible tokens. Each token has a unique **ID**.

Key idea: A token ID is an address; its numerical magnitude does not encode meaning.

[1] Illustrative token split and IDs, chosen for teaching; not the output of a specific GPT tokenizer. ␣marks a space. The actual split depends on the tokenizer.

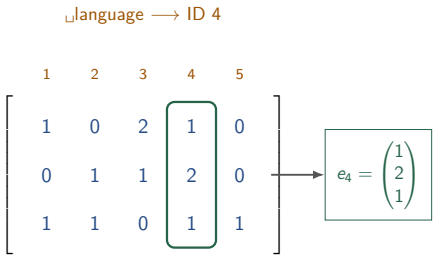
Embedding: a token ID selects a learned column

An **embedding** assigns a continuous numerical vector to a discrete token.

$$W_E \in \mathbb{R}^{d \times V},$$
$$e_i = W_E[:, i]$$

V: vocabulary size; d: vector dimension.

$[:, i]$ means all rows, column i.

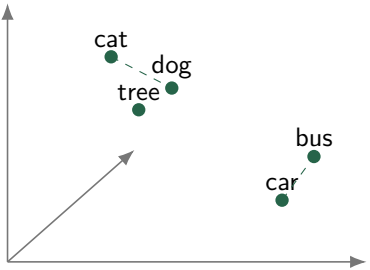


Embedding matrix W_E

Key idea: The table is learned during training. A lookup returns the selected column as an **activation**.

[1] Supplementary toy model: $d = 3$, $V = 5$. Equivalently, $e_i = W_E u_i$, where u_i is a one-hot vector.

Embedding space: proximity can express similarity



The **embedding space** is the vector space of token representations.

Semantic similarity: terms occur with similar meanings or in similar contexts.

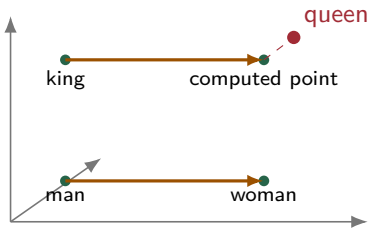
Training can encode these patterns in distances and directions.

Key idea: This is a **constructed 3D illustration**. Real projections lose information.

[1] Proximity depends on the model and metric. Individual axes need not have an interpretable meaning.

Semantic directions: arithmetic with relationships

$$v_{\text{queen}} \approx v_{\text{king}} - v_{\text{man}} + v_{\text{woman}}$$



A **difference vector** is the arrow from one term to another.

$$r = v_{\text{woman}} - v_{\text{man}}$$

Apply the same displacement to v_{king}

A similar idea for singular/plural:

$$v_{\text{cats}} - v_{\text{cat}} \approx v_{\text{dogs}} - v_{\text{dog}}.$$

[1] Illustrative approximations, not universal laws. The video also notes deviations for king/queen. The plotted points are constructed, not measured embeddings; language data may contain stereotypes

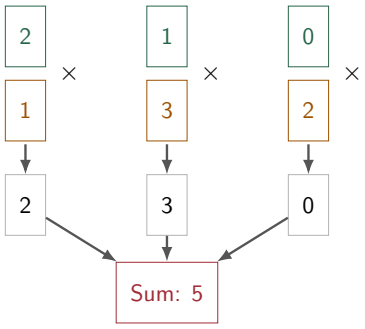
Dot product: multiply pairs, then add

The **dot product** takes two vectors of the same dimension and returns one number.

$$a \cdot b = \sum_{i=1}^{n} a_i b_i$$

$$a = (2, 1, 0),$$
$$b = (1, 3, 2)$$

$$a \cdot b = 2 \cdot 1 + 1 \cdot 3 + 0 \cdot 2$$
$$= 5.$$

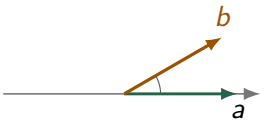


Key idea: Each row of a matrix-vector product computes a dot product with the input vector.

[1] Supplementary numerical example. The output dimension of Wx equals the number of row-wise dot products.

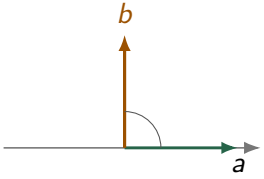
The dot product measures alignment and magnitude

$$a \cdot b = \|a\| \, \|b\| \cos\theta$$



similar directions

$a \cdot b = +0.87$



perpendicular

$a \cdot b = 0$



opposite directions

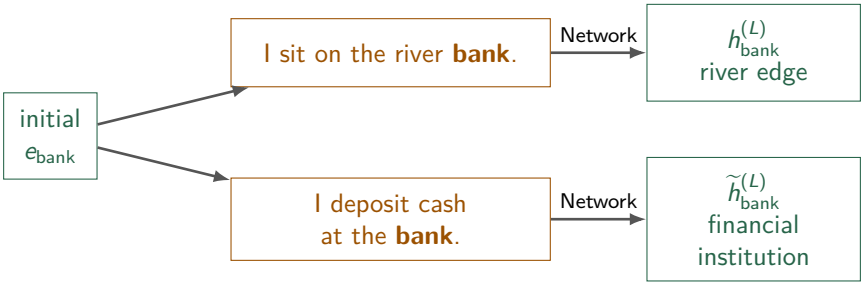
$a \cdot b = -1$

Shown: unit vectors, $\|a\| = \|b\| = 1$.
In general, **vector lengths also affect** the result.

Cosine similarity isolates direction:
$\cos\theta = (a \cdot b)/(\|a\|\|b\|)$ for $a, b \neq 0$.

[1] Connection to attention: dot products of learned projections produce compatibility scores. Details are covered in Chapter 6 of the video series.

Same token – different representations in context

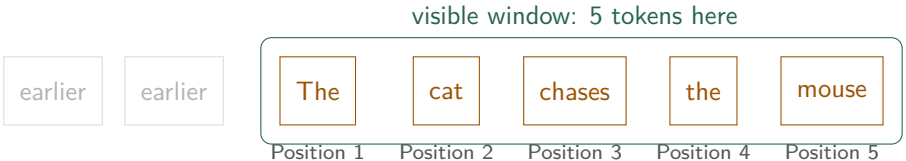


$$e_i \longrightarrow h_i^{(1)} \longrightarrow h_i^{(2)} \longrightarrow h_i^{(L)}$$

Key idea: A **contextual vector** represents the token at this position in the text, beyond its dictionary entry.

[1] Constructed example: assume the same token and ID. Position also affects the representation. Context provides clues but does not guarantee the correct interpretation.

Context and position: why does order matter?



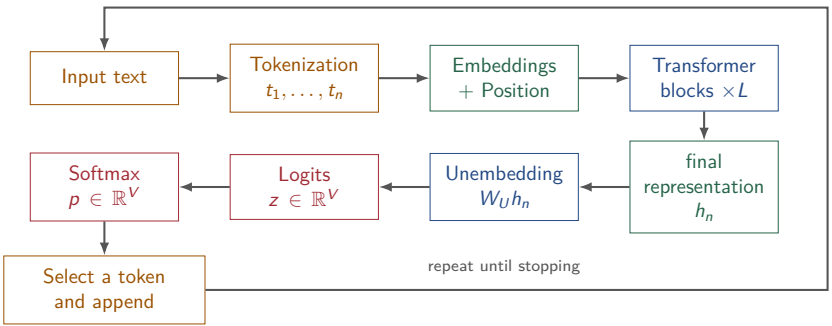
Context Size: maximum number of token positions considered. Excluded text is not directly available.

Position distinguishes, for example: "The cat chases the mouse" and "The mouse chases the cat".

$$h_i^{(0)} = e_{t_i} + p_i \qquad \text{(example: additive position vectors)}$$

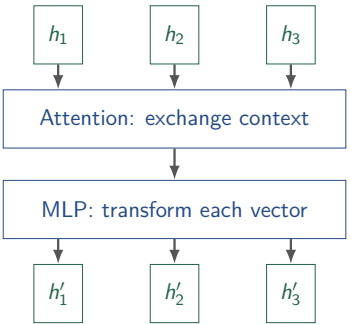
[1, 2] Supplement: additive positional embeddings are one option, not a universal formula for all Transformers. The diagram shows a simplified window. The application selects which text enters the context.

The complete data flow of a GPT language model



[1] Schematic GPT architecture. The representation at the last position n predicts the next token. Colours: orange = tokens; blue = parameters; green = representations; red = scores and probabilities.

Two recurring tasks inside a Transformer block



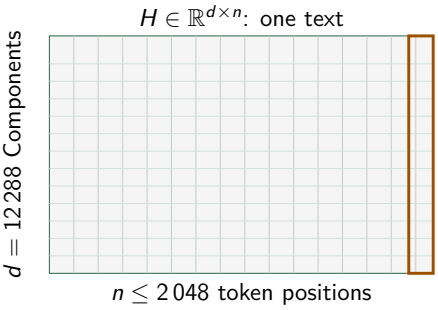
Attention: incorporate relevant earlier positions or the current position.

MLP, multilayer perceptron: apply the same nonlinear transformation to each position, without direct exchange between positions.

Repeated blocks produce increasingly contextual representations.

[1, 2] Schematic overview following Chapter 5; query, key and value are not derived here.
Supplement: GPT uses causal masking to prevent access to future tokens. Residual connections and normalization are omitted for clarity.

From toy examples to the scale of GPT-3



Historical example: GPT-3 (175B)

Vocabulary: $V = 50\,257$ Tokens.
Vector dimension: $d = 12\,288$.
Context: 2 048 Tokens.
Total: about 175 billion parameters.

$$\underbrace{50\,257 \cdot 12\,288}_{\text{Entries in } W_E} = 617\,558\,016$$

Key idea: Parameters are learned tables. The **activations** in H depend on the input text.

[1, 3] Figures: 3Blue1Brown, Chapter 5; Brown et al. (2020), GPT-3 175B. These are not specifications of current ChatGPT models.
Supplement: embedding and unembedding can share weights; matching shapes need not imply extra independent parameters.

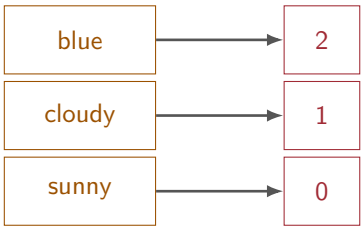
Unembedding: one score for each possible token

$$z = W_U h, \qquad \underbrace{\begin{pmatrix} 2 \\ 1 \\ 0 \end{pmatrix}}_{\text{Logits}} = \underbrace{\begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix}}_{3\times 2} \underbrace{\begin{pmatrix} 2 \\ 1 \end{pmatrix}}_{\text{last representation}}$$

h: final contextual vector at the last input position.

$W_U \in \mathbb{R}^{V \times d}$: one row per vocabulary entry.

Logits: unnormalized scores, not yet probabilities.



[1] Constructed output example: from here, $d = 2$, $V = 3$, separate from the embedding example. Bias omitted. Logits may be negative or exceed 1; their sum is unrestricted. Unembedding is not an inverse embedding operation.

Softmax: make scores positive, then normalize

$$p_i = \frac{e^{z_i}}{\sum_j e^{z_j}}$$

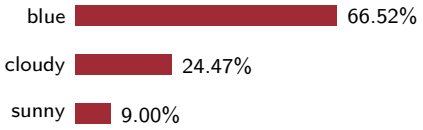
Logits	Exponentiate	Divide by 11.107
2 1 0	7.389 2.718 1	0.6652 0.2447 0.0900

Why e^z? Always positive, strictly increasing and smooth. Score differences become ratios:

$$\frac{p_i}{p_j} = e^{z_i - z_j},$$

$$\sum_i p_i = \frac{\sum_i e^{z_i}}{\sum_j e^{z_j}} = 1.$$

This is a suitable choice, though other normalizations are possible.



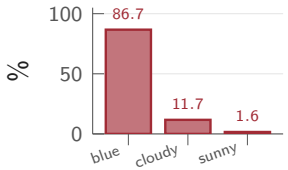
[1] Q...

...

Temperature controls the concentration of the distribution

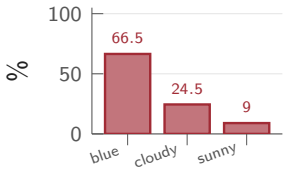
$$p_i(T) = \frac{e^{z_i/T}}{\sum_j e^{z_j/T}}, \qquad T > 0, \qquad z = (2, 1, 0)$$

$T = 0.5$



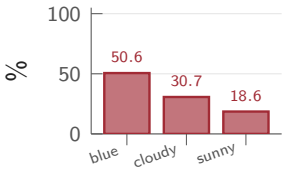
Low: more concentrated, less diverse.

$T = 1$



One: original softmax distribution.

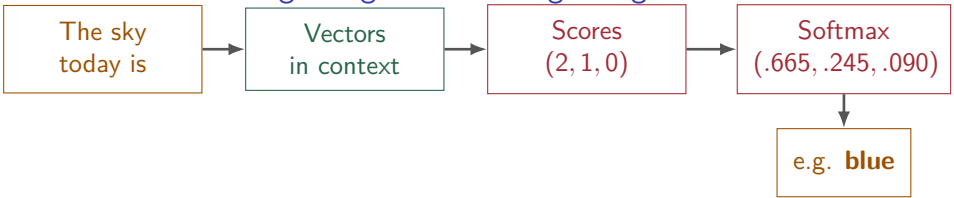
$T = 2$



High: flatter distribution, more randomness.

[1] Own calculation. Greater diversity does not guarantee creativity or correctness. As $T \to \infty$, the distribution becomes uniform.
With a unique maximum, all probability concentrates there as $T \to 0^+$. The setting "$T = 0$" uses argmax selection, not division by zero.

Putting it together: training and generation



Training changes weights.
The actual next token is the target; a low target probability incurs a higher loss:

$$L = -\log p(\text{target token}),$$
$$-\log(0.6652) \approx 0.408.$$

Generation changes the text.
During standard inference, weights stay fixed. Sample a token, append it and process the updated context.

Key idea: Tokens become vectors, context transforms vectors, and scores become probabilities.

[1] Supplement: simplified training loss using the natural logarithm; averaged over many positions. During training, many next-token predictions can be computed in parallel; causal masking hides their target tokens.

References

[1] Sanderson, G. (3Blue1Brown) (2024).
Transformers, the tech behind LLMs – Deep Learning Chapter 5.
Video, 1 April 2024. youtu.be/wjZofJX0v4M.
Companion article: 3blue1brown.com/lessons/gpt/.

[2] Vaswani, A., et al. (2017).
Attention Is All You Need.
arXiv:1706.03762. doi:10.48550/arXiv.1706.03762.

[3] Brown, T. B., et al. (2020).
Language Models are Few-Shot Learners.
arXiv:2005.14165. doi:10.48550/arXiv.2005.14165.

Online sources accessed 17 November 2026. Diagrams and explicitly marked numerical examples: own illustrations.