The weight of how much a token is related to the classes we have defined to understand an input (example of classes are: color, weight, shape, animal, human, liquid,...)

👉 *“How do we know how much a token contributes (or relates) to a particular class we’ve defined, like color, animal, liquid, etc.?”*

This is the link between **token embeddings** (matrix output) and **classification tasks**. Let’s break it down.

**🔎 Step 1: Model Output Matrix (per-token embeddings)**

For your sentence, bert-base-uncased gives:

(sequence\_length × hidden\_size) → (12 × 768)

Each token = a vector of 768 numbers.

**🔎 Step 2: Adding a Classification Layer**

To connect tokens to **classes** (e.g., color, shape, human), we add a **trainable linear layer** on top:

logits = H. W^T +b

* H = the hidden representations (from BERT).
* W = weight matrix, size (num\_classes × hidden\_size).
* b = bias term.

If you have 6 classes (color, weight, shape, animal, human, liquid):

* W has shape **(6 × 768)**.
* Each row in W represents how strongly each hidden dimension maps to that class.

**🔎 Step 3: Token → Class Scores**

For a given token vector h (size 768):

* Multiply by W.
* Output is a vector of size 6 = scores for each class.
* After softmax → probabilities.

Example (fake numbers for token "dog"):

[ color: 0.02, weight: 0.01, shape: 0.05, animal: 0.87, human: 0.03, liquid: 0.02 ]

→ "dog" is highly related to animal.

**🔎 Step 4: Weight Interpretation**

* The **weights in W** tell us how each hidden dimension contributes to each class.
* The **dot product h·W** tells us how much this token activates a given class.
* Training adjusts W so that tokens map correctly to your defined categories.

✅ In short:

* **Tokens → embeddings (matrix).**
* **Classification layer (linear + softmax) → maps embeddings to classes.**
* The **weights in that layer** represent how much features of a token contribute to a specific class.

Would you like me to **draw a diagram** showing how a token embedding flows into a classification head and outputs class probabilities?

**You said:**

yes

**ChatGPT said:**

Perfect 🙌 — here’s a clean diagram of how **token embeddings → class weights → probabilities** works:

**🎨 Flow: Token to Classes**

[Token: "dog"]

│

▼

┌───────────────────────┐

│ Embedding Vector (h) │ size = 768

└───────────────────────┘

│

▼

┌────────────────────────────────────────────┐

│ Classification Weights (W) │

│ Shape = [num\_classes × hidden\_size] │

│ Example = [6 × 768] for {color, weight, │

│ shape, animal, human, liquid} │

└────────────────────────────────────────────┘

│

▼

┌───────────────────────┐

│ Linear Projection │

│ logits = h · Wᵀ + b │ size = num\_classes

└───────────────────────┘

│

▼

┌───────────────────────┐

│ Softmax Activation │

│ Converts logits → │

│ probabilities │

└───────────────────────┘

│

▼

┌───────────────────────────────────┐

│ Probabilities: │

│ color: 0.02 │

│ weight: 0.01 │

│ shape: 0.05 │

│ animal: 0.87 ✅ │

│ human: 0.03 │

│ liquid: 0.02 │

└───────────────────────────────────┘

🔑 Key takeaways:

* **Matrix W** learns *which embedding dimensions* matter for each class.
* **Dot product (h·Wᵀ)** measures how aligned a token is with each class.
* **Softmax** gives normalized probabilities.