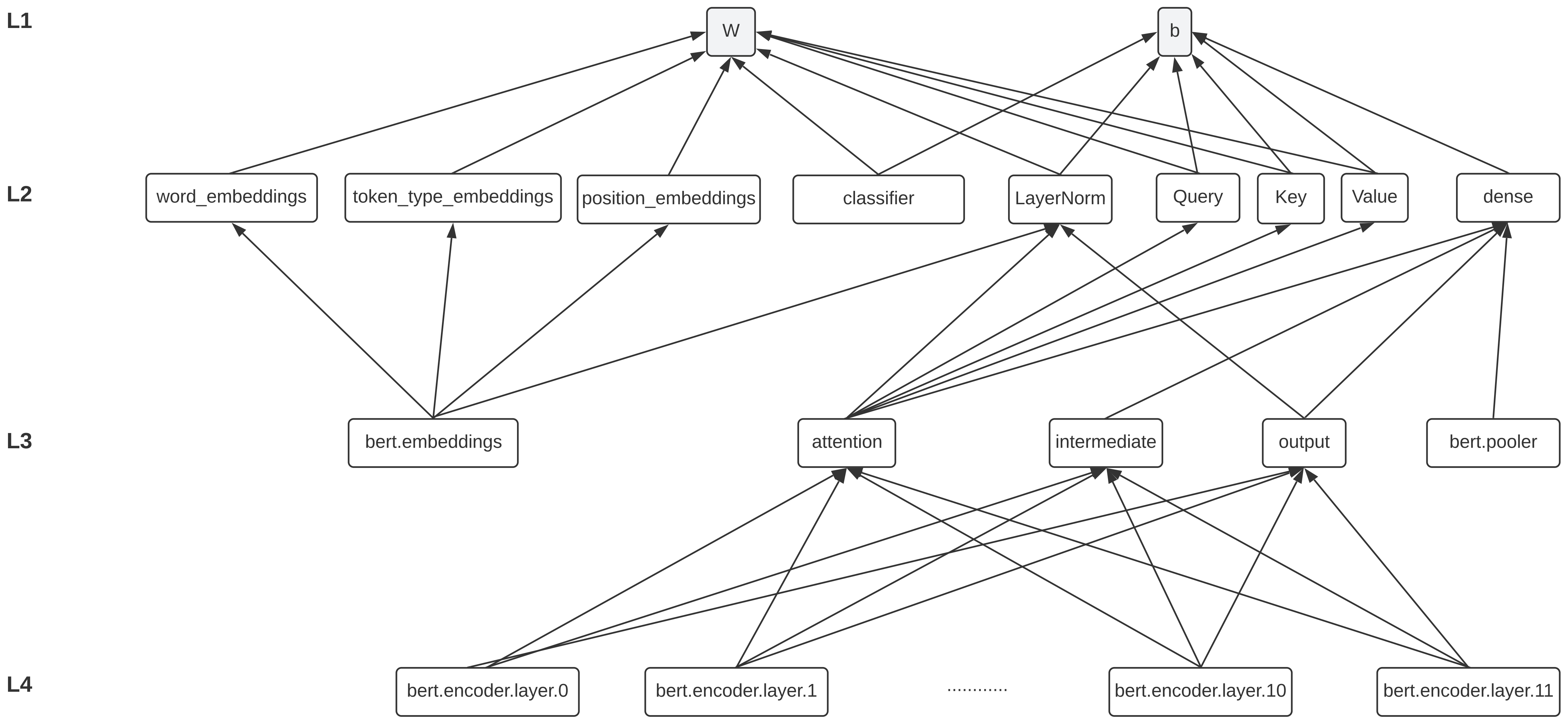
Hierarchial SAID

Abishek S ee18b001 | January 13, 2022



MODEL LAYERS IN PYTORCH

bert.embeddings.word_embeddings.weight
bert.embeddings.position_embeddings.weight
bert.embeddings.token_type_embeddings.weight
bert.embeddings.LayerNorm.weight
bert.embeddings.LayerNorm.bias
bert.encoder.layer.0.attention.self.query.weight
bert.encoder.layer.0.attention.self.query.bias
bert.encoder.layer.0.attention.self.key.weight
bert.encoder.layer.0.attention.self.key.bias
bert.encoder.layer.0.attention.self.value.weight
bert.encoder.layer.0.attention.self.value.bias
bert.encoder.layer.0.attention.output.dense.weight
bert.encoder.layer.0.attention.output.dense.bias
bert.encoder.layer.0.attention.output.LayerNorm.weight
bert.encoder.layer.0.attention.output.LayerNorm.bias
bert.encoder.layer.0.intermediate.dense.weight
bert.encoder.layer.0.intermediate.dense.bias
bert.encoder.layer.0.output.dense.weight
bert.encoder.layer.0.output.dense.bias
bert.encoder.layer.0.output.LayerNorm.weight
bert.encoder.layer.0.output.LayerNorm.bias
...
...
(for 12 such encoder layers)
...
..
bert.pooler.dense.weight
bert.pooler.dense.bias
classifier.weight
classifier.bias

POINTS

We draw the graph like a Hasse Diagram where an edge between two nodes (representing two submodules) denotes the nearest deeper submodule in the model

Total Number of paths in graph to the top level nodes (**W** and **b**) = Number of layers = 201 for mBert

The 3 level H-SAID leverages this hierarchial structure of the model (PyTorch uses this kind of naming, but even otherwise this hierarchy is logically inherent in the model since it has many instances of a block, within which there are modules and so on)

Every path to the top nodes gives rise to a layer and hence a multiplier in SAID. If we partition the graph into two or more components, only the paths within a component from each of the lower level nodes to higher nodes result in a multiplier.

In level 3 hierarchy we partition the graph into L1+L2, L3, L4.

There are a few design choice based on number of partitions and whether we need to take into account some submodule or choose to skip it (for example, we can ignore attention node, and treat attention and intermediate nodes as the same, but having all the logical submodules as node is better for the structure awareness that made SAID successful).