tensor.label_encoder

```

Copy fn label_encoder(self: @Tensor, default_list: Option>, default_tensor: Option>, keys: Option>, keys_tensor: Option>, values: Option>, values_tensor: Option>) -> Tensor;

```

Maps each element in the input tensor to another value.

The mapping is determined by the two parallel attributes, 'keys_' and 'values_' attribute. The i-th value in the specified 'keys_' attribute would be mapped to the i-th value in the specified 'values_' attribute. It implies that input's element type and the element type of the specified 'keys_' should be identical while the output type is identical to the specified 'values_' attribute.

Args

- self
- (@Tensor
-) - The input tensor.
- default_list
- (Option>
-) - The default span.
- default_tensor
- (Option>
-) - The default tensor.
- keys
- (Option>
-) - The keys span.
- keys_tensor
- (Option>
-) - The keys tensor.
- values
- (Option>
-) - The values span.
- values_tensor
- (Option>
-) - The values tensor.
-

One and only one of 'default_'s should be set One and only one of 'keys 's should be set One and only one of 'values*'s should be set.

Panics

- Panics if the len/shape of keys and values are not the same.
-

Returns

A newTensor which maps each element in the input tensor to another value..

Type Constraints

- T
- in (Tensor
- ,Tensor
- ,Tensor
- ,tensor,
-)
-

Examples

```

Copy usearray::{ArrayTrait,SpanTrait}; useorion::operators::tensor::U32Tensor; useorion::operators::tensor::{TensorTrait,Tensor,U32Tensor};
```

```
fnlabel_encoder_example()->Tensor, { fndata()->Tensor { letmutsizes=ArrayTrait::new(); sizes.append(2); sizes.append(3); letmutdata=ArrayTrait::new(); data.append(1); data.append(2); data.append(3); data.append(1); data.append(4); data.append(5);

lettensor=TensorTrait::::new(sizes.span(), data.span()); returntensor; }

fnkeys()->Tensor { letmutsizes=ArrayTrait::new(); sizes.append(3); sizes.append(1);

letmutdata=ArrayTrait::new(); data.append(1); data.append(2); data.append(1);

lettensor=TensorTrait::::new(sizes.span(), data.span()); returntensor; }

fnvalues()->Tensor { letmutsizes=ArrayTrait::new(); sizes.append(3); sizes.append(1);

letmutdata=ArrayTrait::new(); data.append(8); data.append(9); data.append(7);

lettensor=TensorTrait::::new(sizes.span(), data.span()); returntensor; }

fndefault()->Tensor { letmutsizes=ArrayTrait::new(); sizes.append(1);

letmutdata=ArrayTrait::new(); data.append(999);

lettensor=TensorTrait::::new(sizes.span(), data.span()); returntensor; }

letdata=data(); letkeys=keys(); letvalues=values(); letdefault=default(); returndata.label_encoder(default_list:Option::None, default_tensor:Option::Some(default), keys:Option::None, keys_tensor:Option::Some(keys), values:Option::None, values_tensor:Option::Some(values));

                    [7,9,999,7,999,999],
```

Previous tensor.random_uniform_like Next Neural Network

Last updated15 days ago