Performing Multi-class Text Classification Using Characters

Janani Ravi
CO-FOUNDER, LOONYCORN
www.loonycorn.com

Overview

RNN models that operate on characters rather than on words

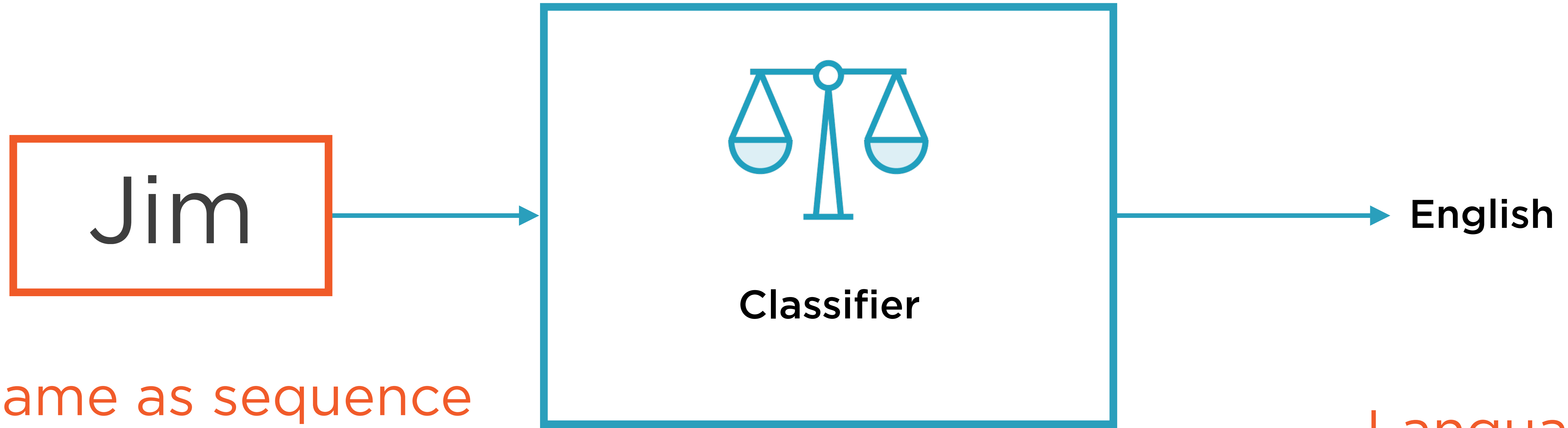
Represent characters as tensors

Use dynamic computation graphs to cope with differing word lengths

Multi-class text classification using RNNs

Language Prediction Based on Names

Language Prediction Based on Names

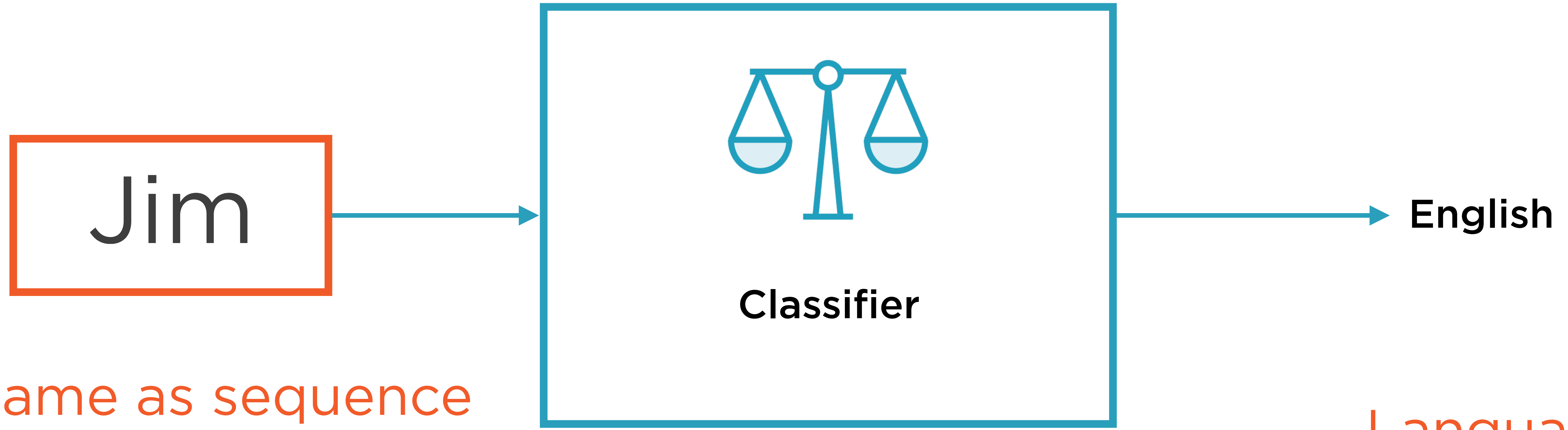


Name as sequence of characters

Language

In PyTorch, if we use dynamic computation graphs we do not need to pad names to be of the same length

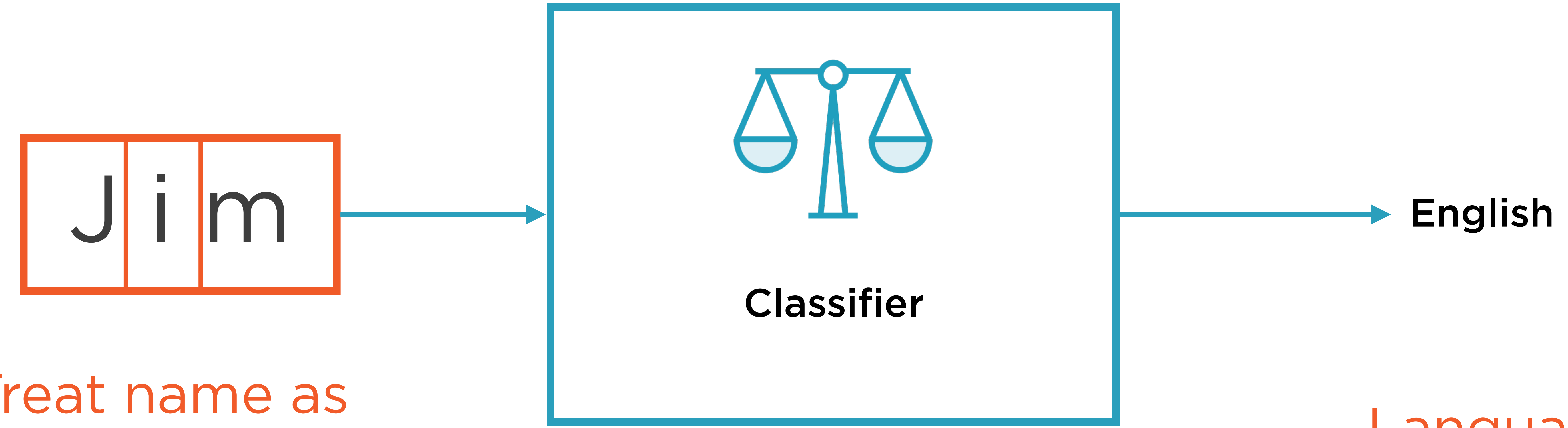
Language Prediction Based on Names



Name as sequence of characters

Language

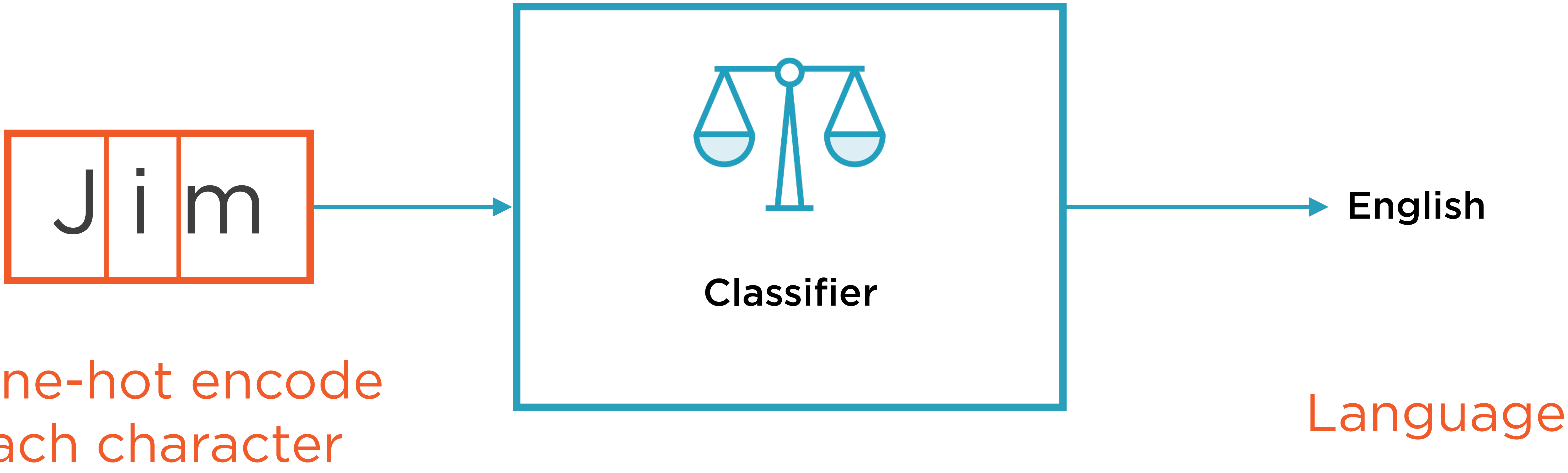
Language Prediction Based on Names



Treat name as sequence of characters

Language

Language Prediction Based on Names



One-hot Encoded Characters

Letter	a	...	j	...	i	...	m	...
J	0	0	1	0	0	0	0	0
i	0	0	0	0	1	0	0	0
m	0	0	0	0	0	0	1	0

26 elements

One-hot Encoded Characters

Letter	a	...	j	...	i	...	m	...
J	0	0	1	0	0	0	0	0
i	0	0	0	0	1	0	0	0
m	0	0	0	0	0	0	1	0

26 elements

One-hot Encoded Characters

Letter	a	...	j	...	i	...	m	...
J	0	0	1	0	0	0	0	0
i	0	0	0	0	1	0	0	0
m	0	0	0	0	0	0	1	0

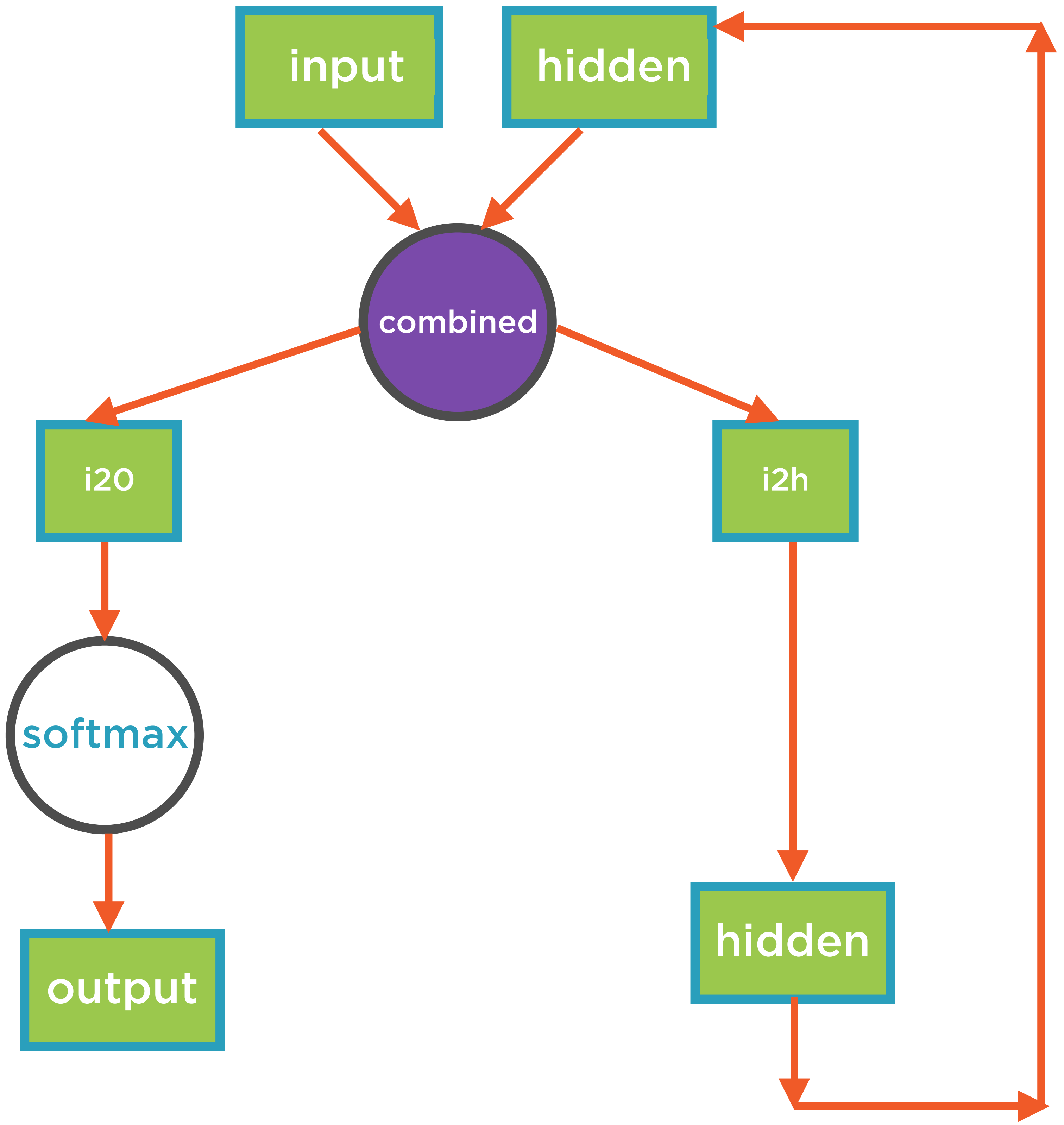
26 elements

One-hot Encoded Characters

Letter	a	...	j	...	i	...	m	...
J	0	0	1	0	0	0	0	0
i	0	0	0	0	1	0	0	0
m	0	0	0	0	0	0	1	0

26 elements

```
class RNN(nn.Module):

# In constructor

  self.i2h = nn.Linear()

  self.i2o = nn.Linear()

  self.softmax = nn.LogSoftmax()

# Forward method

  def forward(self, input, hidden):

        combined = torch.cat((input, hidden), 1)

        hidden = self.i2h(combined)

        output = self.i2o(combined)

        output = self.softmax(output)

        return output, hidden
```



Demo

Multi-class text classification using RNNs

Summary

RNN models that operate on characters rather than on words

Represent characters as tensors

Use dynamic computation graphs to cope with differing word lengths

Multi-class text classification using RNNs