1.	Identify the correct order of the gates that information flows through in an LSTM unit.	1 / 1 point
	O Input gate, forget gate, output gate.	
	Forget gate, input gate, output gate.	
	Output gate, forget gate, input gate.	
	O Forget gate, output gate, input gate	
	Correct.	

1/1 point

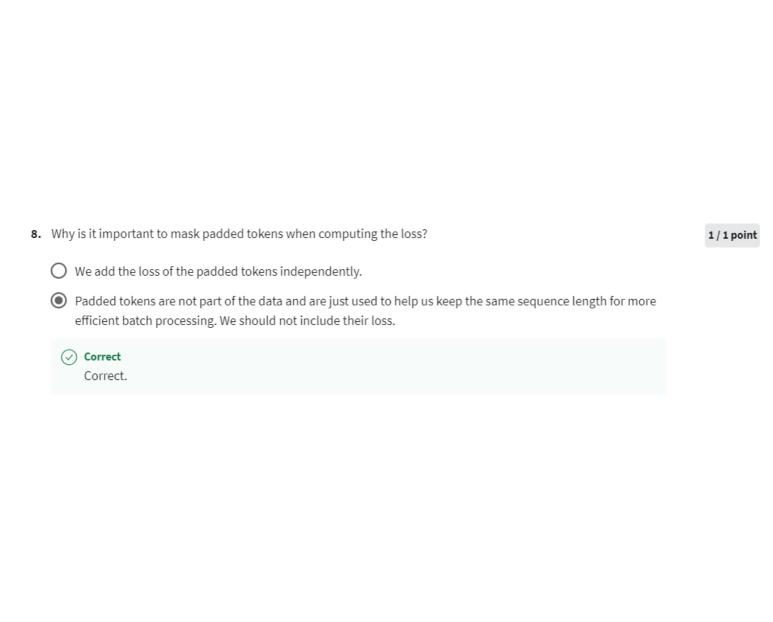
The tanh layer ensures the values in your network stay numerically stable, by squeezing all values between -1 and 1. This prevents any of the values from the current inputs from becoming so large that they make the other values insignificant.	1
<ul><li>○ False</li><li>● True</li></ul>	

What type of architecture is a named entity recognition using?	1 / 1 point
O Many to one	
Many to many	
One to many	
○ Correct     Correct.	
	<ul> <li>Many to one</li> <li>Many to many</li> <li>One to many</li> <li>✓ Correct</li> </ul>

5.	Extract the named entities from the following sentence:	1 / 1 point
	Younes, a Moroccan artificial intelligence engineer, travelled to France for a conference.	
	O Younes, Moroccan, conference.	
	Younes, Moroccan, France.	
	O Younes, Moroccan, engineer.	
	O Younes, Moroccan engineer, France.	
	Correct.	

6.	In a vectorized representation of your data, equal sequence length allows more efficient batch processing.	1 / 1 point
	<ul><li>False</li><li>True.</li></ul>	

7.	Which built-in Python method would you use to iterate over your test set during the evaluation step? Assuming you are using a data generator.	1 / 1 poin
	next()	
	O list()	
	O enumerate()	
	O slice()	
	Correct.	



9.	In wh	nich of the following orders should we train an Named Entity Recognition with an LSTM?
	0	Create a tensor for each input and its corresponding number
		2. Put them in a batch => 64, 128, 256, 512
		3. Run the output through a dense layer
		4. Feed it into an LSTM unit
		5. Predict using a log softmax over K classes
	•	Create a tensor for each input and its corresponding number
		2. Put them in a batch => 64, 128, 256, 512
		3. Feed it into an LSTM unit
		4. Run the output through a dense layer
		5. Predict using a log softmax over K classes
	0	Create a tensor for each input and its corresponding number
		2. Put them in a batch => 64, 128, 256, 512
		3. Run the output through a dense layer
		4. Predict using a log softmax over K classes
		5. Feed it into an LSTM unit
	$\odot$	Correct.

1 / 1 point

10.	LSTMS solve vanishing/exploding gradient problems when compared to basic RNNs.
	<ul><li>False</li><li>True</li></ul>
	Correct.

1 / 1 point