1a.

Stack	Buffer	New Dependency	Transition
[ROOT]	[Nadia, rode, the, old, donkey, with, dexterity]		Initial Config
[ROOT, Nadia]	[rode, the, old, donkey, with, dexterity]		SHIFT
[ROOT, Nadia, rode]	[the, old, donkey, with, dexterity]		SHIFT
[ROOT, rode]	[the, old, donkey, with, dexterity]	rode –nsubj-> Nadia	LEFT-ARC
[ROOT, rode, the]	[old, donkey, with, dexterity]		SHIFT
[ROOT, rode, the, old]	[donkey, with, dexterity]		SHIFT
[ROOT, rode, the, old, donkey]	[with, dexterity]		SHIFT
[ROOT, rode, the, donkey]	[with, dexterity]	donkey –amod-> old	LEFT-ARC
[ROOT, rode, donkey]	[with, dexterity]	donkey –det-> the	LEFT-ARC
[ROOT, rode]	[with, dexterity]	rode –dobj-> donkey	RIGHT-ARC
[ROOT, rode, with]	[dexterity]		SHIFT
[ROOT, rode, with, dexterity]			SHIFT
[ROOT, rode, with]		with –pobj-> dexterity	RIGHT-ARC
[ROOT, rode]		rode –prep-> with	RIGHT-ARC
[ROOT]		ROOT-root -> rode	RIGHT-ARC

1b. the number of steps it will take is 2n + 1. For each word in the sentence, it will take one step to add the word to the stack and one step to remove the word from the stack. The added 1 is for initializing the commit.

1c. The reason can be shown using the example that is given.

Stack	Buffer	New Dependency	Transition
[ROOT]	[John, saw, a dog, yesterday, which, was, a Yorkshire Terrier]		Init Config
[ROOT, John]	[saw, a dog, yesterday, which,		SHIFT

	was, a Yorkshire Terrier]		
[ROOT, John, saw]	[a dog, yesterday, which, was, a Yorkshire Terrier]		SHIFT
[ROOT, saw]	[a dog, yesterday, which, was, a Yorkshire Terrier]	saw –nsubj-> John	LEFT-ARC
[ROOT, saw, a dog]	[yesterday, which, was, a Yorkshire Terrier]		SHIFT
[ROOT, saw, a dog, yesterday]	[which, was, a Yorkshire Terrier]		SHIFT
[ROOT, saw, a dog, yesterday, which]	[was, a Yorkshire Terrier]		SHIFT
[ROOT, saw, a dog, yesterday, which, was]	[a Yorkshire Terrier]		SHIFT
[ROOT, saw, a dog, yesterday, was]	[a Yorkshire Terrier]	was –nsubj-> which	LEFT-ARC
[ROOT, saw, a dog, yesterday, was, a Yorkshire Terrier]	[]		SHIFT
[ROOT, saw, a dog, yesterday, was]	[]	was –attr-> a Yorkshire Terrier	RIGHT-ARC

From here you see the error. There should be a RIGHT-ARC from “a dog” to “which” and “saw” to “yesterday”. However, that cannot happen because neither the two sets are directly adjacent to each other in the stack. As such the program becomes stuck.
An attempt to fix this might be to backtrack to “a dog” and RIGHT-ARC instead of SHIFT.

[ROOT, John, saw]	[a dog, yesterday, which, was, a Yorkshire Terrier]		SHIFT
[ROOT, saw]	[a dog, yesterday, which, was, a Yorkshire Terrier]	saw –nsubj-> John	LEFT-ARC
[ROOT, saw, a dog]	[yesterday, which, was, a Yorkshire Terrier]		SHIFT
[ROOT, saw]	[yesterday, which, was, a Yorkshire Terrier]	saw –dobj-> a dog	RIGHT-ARC

[ROOT, saw, yesterday]	[which, was, a Yorkshire Terrier]		SHIFT
[ROOT, saw]	[which, was, a Yorkshire Terrier]	saw –npadvmod-> yesterday	RIGHT-ARC

.
However, by doing so “a dog” gets taken out of the stack and now cannot have a RIGHT-ARC to “was”.

2b

Strategy:

My plan was as follows. First, I’d create some place holders that can later be fed data during training. (*add_placeholders)* Then I would create a feed dictionary that would map the place holders to their respective values. This feed dictionary could then later be used during training for mapping. (*create_feed_dict*) After, using my Xavier Initalizer, I would create new matrices that consist of all embeddings for all given ids. And then I can create look up tables by utilizing emedding_lookup to grab a subset of the matrices that I had just initialized. By doing so I am able to convert words into their appropriate vector forms. (*Add_embeddings*)

Then, with my matrixes set up I can add the layer for softmax using the provided formulas. (*add_prediction_op*) After that it becomes a simple matter of calculating how correct the predictions were (*add_loss_op*) and adding the training operations. (*add_training_op*).

Difficulties:

Overall the implementation was fairly straightforward. Some difficulties arose in add_embeddings when I made the mistake of thinking that the embeddings were the ids and the placeholders were the params. Another small error occurred in add_prediction_op where some issues came up in tensor multiplication surrounding the multiply function and matching the dimensions of the shapes. This was solved though some matrix multiplication and using tf.matmul instead of tf.multiply.

A larger issue I had was when the program started hanging on the first epoch. I took me a while to realise this was because the mini_batch function was not working correctly.

The biggest issue that I had, and the one I was regrettably unable to fix, is in regard to my test LAS and test UAS. The best scores I could get were 0.01 and 0.08 respectively. This issue can mostly be narrowed down to either improperly initialized variables or an issue in parser.py. Using the advice on the forum I focused my attention on fixing the oracle function and getting the training data to some value closer to my peers. In the end, I was able to narrow it down to 1892478 samples. However, this fix did not seem to have any effect on the LAS or UAS score. I have also tried to see whether it was an issue with improper initialization but was not able to find the source.

However, my cross-entropy readings do show a decrease which means that there are some predictions being done.