First, I would like to point out the most challenging part that I encountered during my trials, which python said the data can not be decoded. This refers to "'utf-8' codec can't decode byte exc3 in position @: unexpected end of data". I searched stackflow, though there are several solutions related to this problem, but no one works. Finally, I solved this problem buy changing the encoding to ANSI which works better when deal with English letter. Hope this can be helpful if later students have the same issue (I bet this could only happen when you run code on your local computer).

I formed 6 sets of trials in this homework.

This is the first set of my trials.

# set	# model	test accuracy	time	size(train/valid/test)	max length	max token	naïve(RNN-LSTM)	embedding	embedding with masking	pretrained model(GloVe)
1	1	0.841	5m15s	20000/5000/25000	150	20000	YES			
	2	0.835	3m6s	20000/5000/25000	150	20000		YES		
	3	0.838	3min25s	20000/5000/25000	150	20000			YES	
	4	0.824	2m55	20000/5000/25000	150	20000				YES

Initially, I was trying to replicate the results from textbook first which required the max length = 600, however, my gpu reported OOM issue so that I cut it to 150 as the assignment required then later sets of trials will enlarge this number. For the other model settings as size, token, etc., I followed the textbook.

As you can see from the table above, with sufficient training size, the LSTM model(naïve one) performed best among the four different models. I would not say that there is a huge difference, because there is a trade-off between accuracy and time of training. Standing at this point, I would still prefer to choose the naïve model, because I know under the condition of having a large available training set, embedding and GloVe will not help too much.

This is my second set of trials.

2	1	0.575	1m29s	100/10000/25000	150	20000	YES			
	2	0.614	35.8s	100/10000/25000				YES		
	3	0.633	47s	100/10000/25000					YES	
	4	0.571	48.6s	100/10000/25000						YES

In this set, I designed the model based on the requirements from the hw4. Specifically with max length =150 and size as 100/10000/25000. Due to limited training set, I expected that the model with embedding and pretrained model will perform better compared to LSTM(naïve) one.

It turns out that embedding or embedding with masking improved the performance and at the same time, it saved the training time, which is good. However, the pretrained model shows no difference compared to naïve model. It does not make too much sense, and I cannot figure out a reasonable answer for it.

With limited sample, even with embedding and pretrained applied, the test accuracy is still far aways from rich sample. Embedding does help but still limited.

This is my third set of trials.

3	1	0.832	2m33s	10000/10000/25000	150	20000	YES			
	2	0.826	1m25s	10000/10000/25000	150	20000		YES		
	3	0.831	1m49s	10000/10000/25000	150	20000			YES	
	4	0.82	1m51s	10000/10000/25000	150	20000				YES

After doing the first two trials. One question left about why pretrained model did not beat naïve model in the second trial (#2, model 4). My guess is that the training size is too small. Then in this set of trials. I increased the sample to 10000/10000/25000. And my expectation is that in this case, pretrained model should beat naïve model at least in some degrees.

However, my models turn out that the pretrained model still did not beat the naïve model. This could be due to the training sample is large as 10000 and naïve model can map pretty good in this learning. And embedding did not outperform but saved the training time.

This is my fourth set of trials.

4	1			3000/10000/25000	150	20000	YES			
	2			3000/10000/25000	150	20000		YES		
	3			3000/10000/25000	150	20000			YES	
	4			3000/10000/25000	150	20000				YES

Motived by the third set of trials, I decided to reduce the sample size to 3000/10000/25000 to check if embedding and pretrained models can dominate the naïve model.

It turns me down again. Because the pretrained model did not beat the naïve model and only model with embedding and masking beat the naïve model. My guess is that our training sample is still too large to make the difference between naïve and GloVe.

This is my fifth of trials.

5	1	0.704	1m22s	600/10000/25000	150	10000	YES			
	2	0.718	37.5s	600/10000/25000	150	10000		YES		
	3	0.716	1m49s	600/10000/25000	150	10000			YES	
	4	0.645	55.6s	600/10000/25000	150	10000				YES, 100d
	5	0.683	57s	600/10000/25000	150	10000				GloVe 200d
	6	0.715	1m4s	600/10000/25000	150	10000				GloVe 300d

To figure out why GloVe performed badly. I reduced the training sample down to 600. This time, there are some insights from these models. With trying 100d, 200d, 300d of GloVe. It turns out that with limited data, using higher dimension of pretrained model will improve the model accuracy. In my case, it improves the accuracy from 0.704 to 0.715.

This is my last set of trials.

6	1	0.878	7m56s	20000/5000/25000	400	20000	YES			
	2	0.849	4m33s	20000/5000/25000	400	20000		YES		
	3	0.874	4m29s	20000/5000/25000	400	20000			YES	
	4	0.879	4m52s	20000/5000/25000	400	20000				GloVe 300d

In this series, I release the constraints on sample size and length and word to build the best model. The takeaways of this set of trials are that align with the previous experiments, with enriched data, naïve model can be a good choice for accuracy, but it takes longer time to train. An improvement can be using GloVe high dimension model(300d). It can give the same or better result but save a lot of training time.

In conclusion, embedding helps only with the scenario of limited training sample, and it can save time compared to naïve RNN. GloVe can sometimes help with limited training sample, but this is no guarantee. But high D GloVe is the optimal solution once you have a large sample and target on best accuracy and training time.