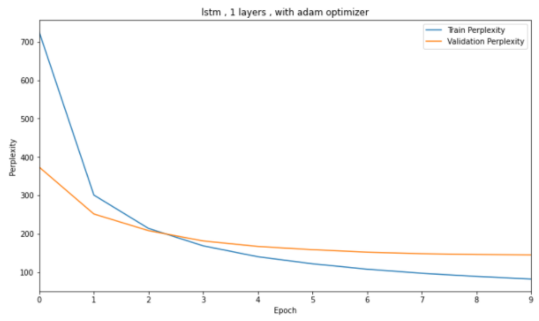
Due Date: March 22nd 2021, 11pm EST

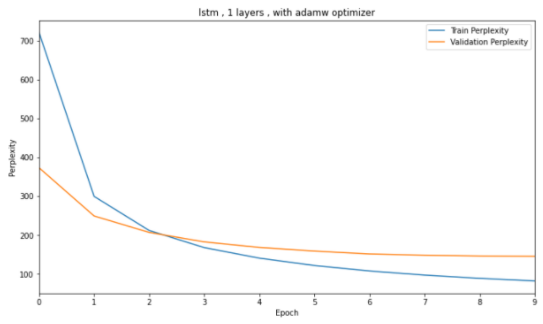
Problem 1

1. You are asked to run 12 experiments with different architectures, optimizers, and hyperparameters settings. These parameter settings are given to you at the top of the runner file (`run_exp.py`). For each of these 12 experiments, plot learning curves (train and validation) of perplexity over both **epochs** and **wall-clock-time**. Figures should have labeled axes and a legend and an explanatory caption.
   Note: For better comparison, each figure corresponds to a model and each sub figure corresponds to an optimizer.

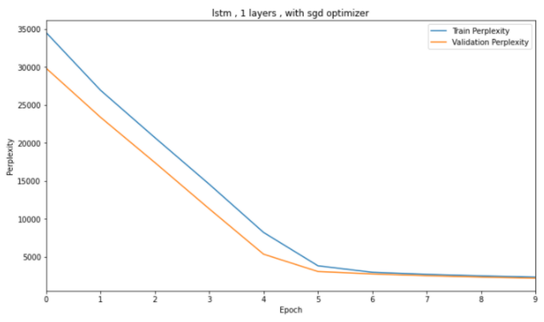
PPL over epochs
LSTM model , 1 layer



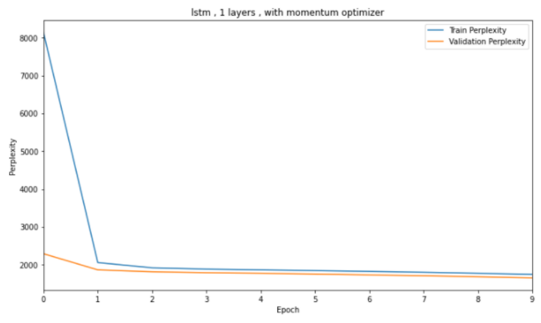
((a)) Adam Optimizer



((b)) AdamW Optimizer



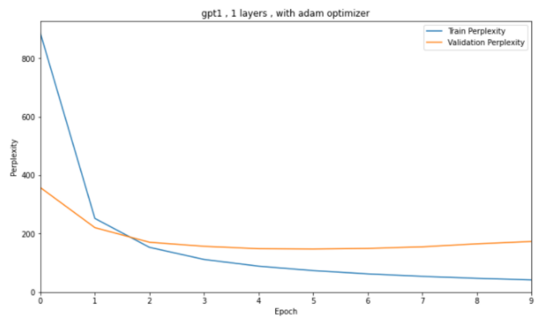
((c)) SGD Optimizer



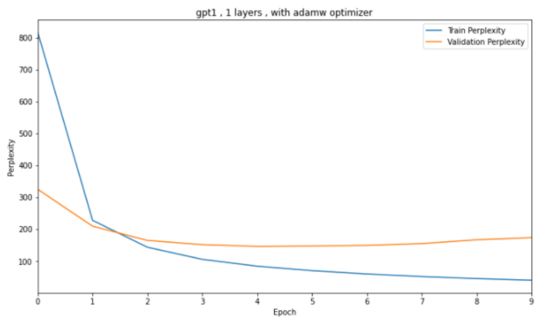
((d)) Momentum Optimizer

Figure 1: PPL over epochs for LSTM model with 1 layer using different optimizers: **(a):Adam Optimizer**, **(b):AdamW Optimizer**,**(c):SGD Optimizer**,**(d):Momentum Optimizer**

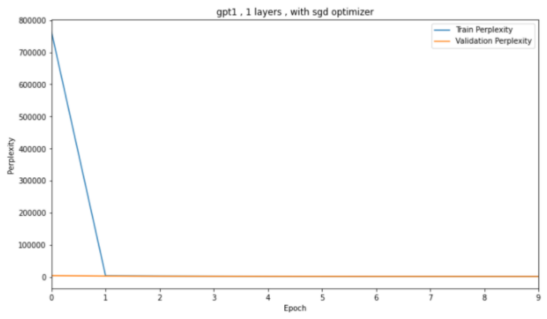
GPT1 model , 1 layer



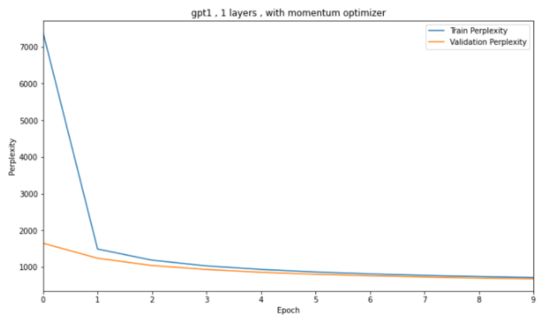
((a)) Adam Optimizer



((b)) AdamW Optimizer



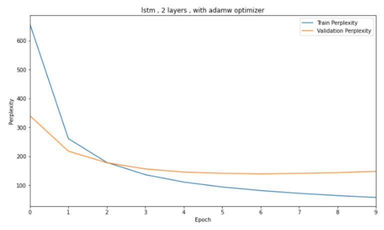
((c)) SGD Optimizer



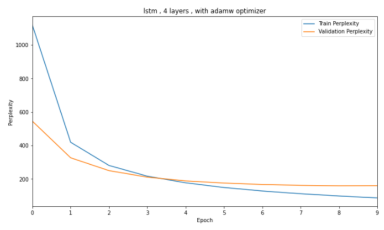
((d)) momentum Optimizer

Figure 2: PPL over epochs for GPT1 model with 1 layer using different optimizers: **(a):Adam Optimizer**, **(b):AdamW Optimizer**,**(c):SGD Optimizer**,**(d):Momentum Optimizer**

LSTM with Adam Optimizer



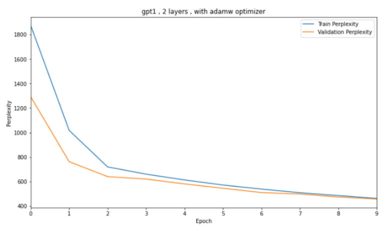
((a)) 2 layers



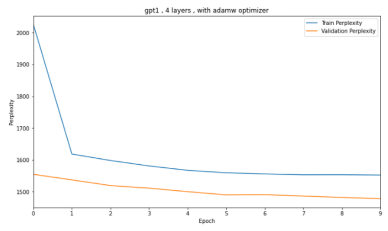
((b)) 4 layers

Figure 3: PPL over epochs for LSTM with Adam Optimizer but with different layerso number: **(a) 2 layers**,**(b) 4 layers**

GPT1 with Adam Optimizer



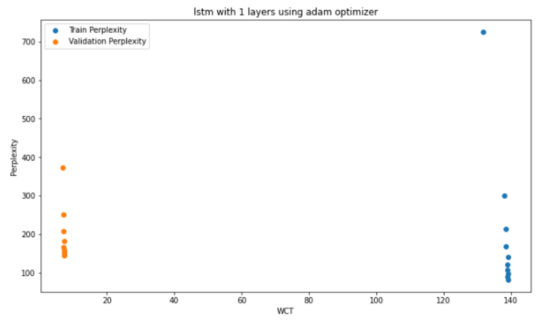
((a)) 2 layers



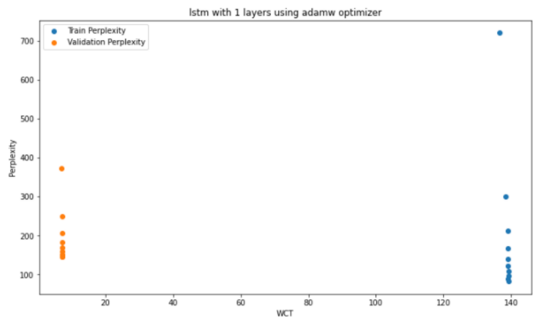
((b)) 4 layers

Figure 4: PPL over epochs for GPT1 with Adam Optimizer but with different layerso number: **(a) 2 layers**,**(b) 4 layers**

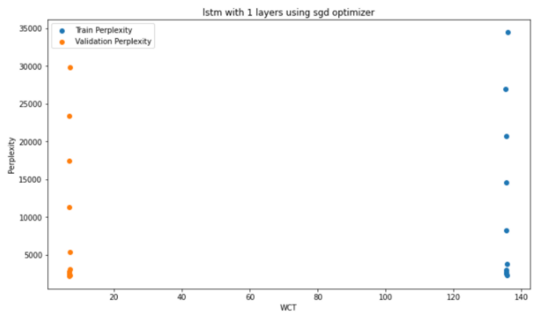
PPL over wall-clock-time
LSTM model , 1 layer



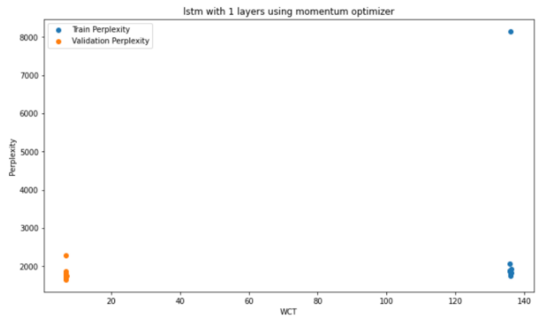
((a)) Adam Optimizer



((b)) AdamW Optimizer



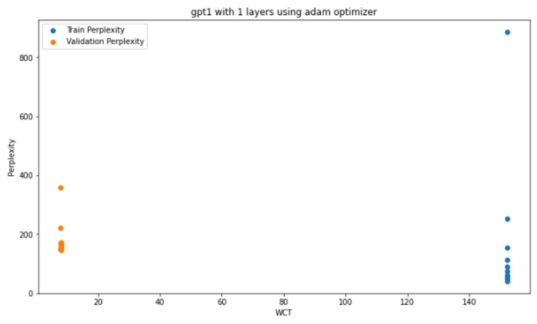
((c)) SGD Optimizer



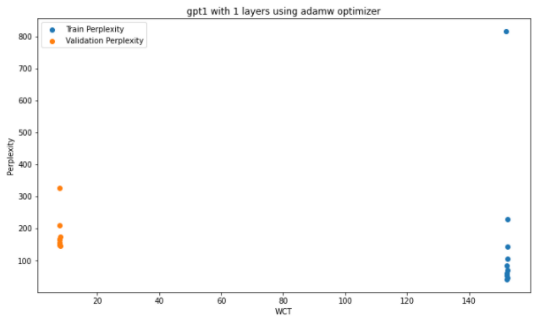
((d)) Momentum Optimizer

Figure 5: PPL over wall-clock-time for LSTM model with 1 layer using different optimizers: **(a):Adam Optimizer**, **(b):AdamW Optimizer**,**(c):SGD Optimizer**,**(d):Momentum Optimizer**

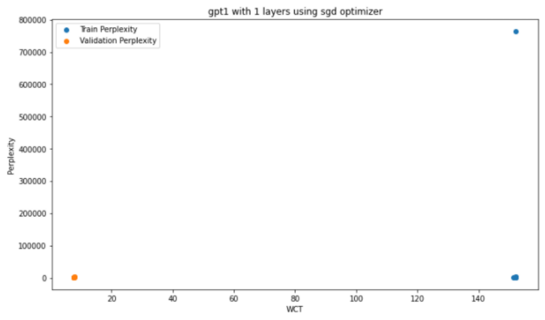
GPT1 model , 1 layer



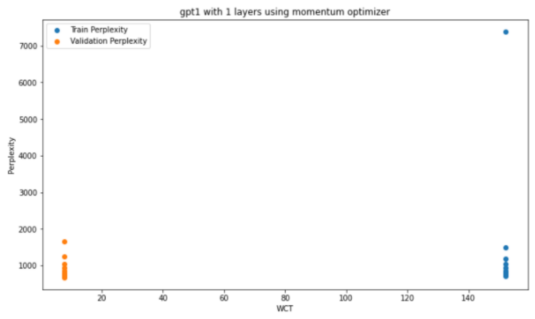
((a)) Adam Optimizer



((b)) AdamW Optimizer



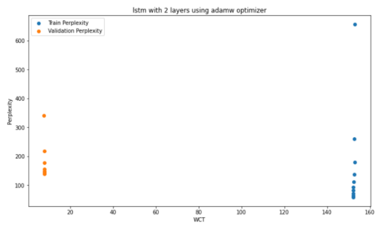
((c)) SGD Optimizer



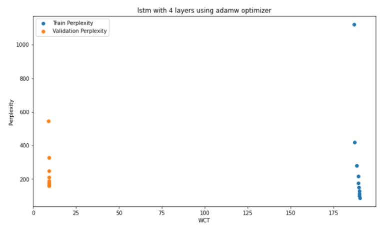
((d)) momentum Optimizer

Figure 6: PPL over wall-clock-time for GPT1 model with 1 layer using different optimizers: **(a):Adam Optimizer**, **(b):AdamW Optimizer**,**(c):SGD Optimizer**,**(d):Momentum Optimizer**

LSTM with Adam Optimizer



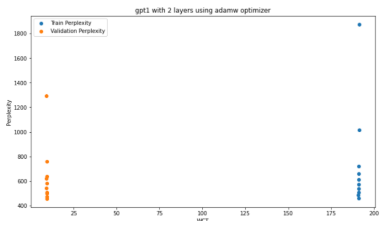
((a)) 2 layers



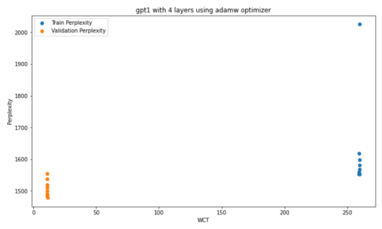
((b)) 4 layers

Figure 7: PPL over wall-clock-time for LSTM with Adam Optimizer but with different layerso number: **(a) 2 layers**,**(b) 4 layers**

GPT1 with Adam Optimizer



((a)) 2 layers



((b)) 4 layers

Figure 8: PPL over wall-clock-time for GPT1 with Adam Optimizer but with different layerso number: **(a) 2 layers**,**(b) 4 layers**

2. Make a table of results summarizing the train and validation performance for each experiment, indicating the architecture and optimizer. Sort by architecture, then number of layers, then optimizer, and use the same experiment numbers as in the runner script for easy reference. Bold the best result for each architecture.[1] The table should have an explanatory caption, and appropriate column and/or row headers. Any shorthand or symbols in the table should be explained in the caption.

Architecture	Number of layers	Optimizer	Train PPL	Validation PPL	Epoch	Configs
GPT1	1	ADAM	73.612	147.398	6	5
GPT1	**1**	**ADAMW**	**84.177**	**146.199**	**5**	**6**
GPT1	1	MOMENTUM	715.642	678.697	10	8
GPT1	1	SGD	1604.716	1491.724	10	7
GPT1	2	ADAMW	462.757	457.414	10	11
GPT	4	ADAMW	1552.178	1478.023	10	12
LSTM	1	ADAM	81.881	144.841	10	1
LSTM	1	ADAMW	81.854	145.261	10	2
LSTM	1	MOMENTUM	1745.266	1652.109	10	4
LSTM	1	SGD	2322.149	2173.797	10	3
LSTM	**2**	**ADAMW**	**81.714**	**139.337**	**7**	**9**
LSTM	4	ADAMW	98.359	159.248	9	10

Table 1: Results of the epoch with the best validation PPL(perplexity), for each experiment. For each config(column Configs), we mention the train and validation PPl that corresponds to an epoch (Epoch column) for both architecture with different layers number and different optimizer. The bold lines (config 6 and 9) corresponds to the best result, i.e. best validation PPL, for each architecture.

[1]You can also make the table in LaTeX; for convenience you can use tools like LaTeX table generator to generate tables online and get the corresponding LaTeX code.

3. Which hyperparameters + optimizer would you use if you were most concerned with wall-clock time? With generalization performance?

   - **Wall-clock time**:
     If we are concerned with wall-clock time, we should focus on figures that describe perplexity over wall-clock-time. As we can notice that figures 7 and 8, which show LSTM and GPT1 performances with 2/4 layers, took alot of time for training. For example GPT1 with 4 layers usin ADAMW optimizer take more than 250 wct unit. In the other hand, GPT1 with 1 single layer, in the figure 6, with all the different optimizer it take more than 140 wct unit. However, LSTM model with 1 layer, in figure 5, WCT is in the range of 140 wct unit (140 wct or less a little bit)the different optimzers.

     LSTM+ 1 layer with different optimizer have the lowest training time, so to choose one of them we should focus on validation time. With ADAM optimizer, we have 6.94 wct unit as a lowest validation time. To conclude, If we are concerned with wall-clock time then we choose LSTM, 1 layer using ADAM optimzers.

   - **Generalization performance**:
     Generalization refers to your model's ability to adapt properly to new, previously unseen data. That is mean the gap between Train PPL and validation PPL should be minimal. Referring to the previous table 1, we notice that the smallest gap betwwen train PPL and validation PPL corresponds to GPT1 2 layers and ADAMW optimizer (train PPL of last epoc = 462.757 and validation ppl = 457.441).

4. Between the experiment configurations 1-4 and 5-8 at the top of `run_exp.py`, only the optimizer changed. What difference did you notice about the four optimizers used? What was the impact of weight decay, momentum, and Adam?
   Referring to the figures 1, and 2, we can notice that with:
   - For **ADAM optimizer** and **AdamW optimizer**, the train and validation PPLs decrease slowly until epoch 2 the train PPL keep decreases but validation PPL stabilizes at a certain value(200) and we notice there is a gap between them. Also, with **ADAM optimizer** and **AdamW optimizer** we get smallest PPL values, as mentionned in the table 1.
   - For **SGD optimizer**, the train and validation PPLs decrease more quickly until the 5th eoch they stabilize at same value. Also, for GPT1, validation PPl is equal to zero.
   - For**Momentum optimizer**, the train PPL decrease very very quickly comparing to others and in the other hand the validation PPL is very slowly and does not decrease but it get stuck for a certain value.

5. Compare experiments 1 and 5. Which model did you think performed better (LSTM or GPT)? Why?
   Comparing experiments 1 and 5, i.e. 1 layer with ADAM optimizer, LSTM performed better than GPT1. By looking at subfigure (a) in figure 1 and subfigure (a) in figure 2, we can clearly see that the gap between train PPl and validation PPl in GPT1 is larger than in LSTM. Therefore, LSTM has a better generalization performance.

6. In configurations 5-8 and in configurations 11 and 12, you trained a transformer with various hyper-parameter settings. Given the recent high profile transformer based language models, are the results as you expected? Speculate as to why or why not.

   Transformer models aim to solve sequence-to-sequence tasks while handling long-range dependencies with ease, see all words simultaneously. Furthermore,they are attention based models. So it was expected to get better results with GPT1 rather than LSTM.
   The performance limitations of GPT1 can be due to the fact we did not use more data, 1,2 or even 4 layers are not enough to get better results. Also, we should mention that transformer can only deal with fixed-length text strings, so this can affect the performance of the model to.

7. For each of the experiment configurations above, measure the average steady-state GPU memory usage (`nvidia-smi` is your friend!). Comment about the GPU memory footprints of each model, discussing reasons behind increased or decreased memory consumption where applicable.

Experiment	Memory-Usage	GPU-Util
LSTM+1layer+Adam	6364MiB / 15109MiB	98%
LSTM+1layer+AdamW	6364MiB / 15109MiB	95%
LSTM+1layer+SGD	6364MiB / 15109MiB	88%
LSTM+1layer+Monmentum	6364MiB / 15109MiB	97%
GPT1+1layer+Adam	6364MiB / 15109MiB	71%
GPT1+1layer+AdamW	6364MiB / 15109MiB	99%
GPT1+1layer+SGD	6364MiB / 15109MiB	95%
GPT1+1layer+Monmentum	6364MiB / 15109MiB	66%
LSTM+2layer+AdamW	5408MiB / 15109MiB	82%
LSTM+4layer+AdamW	6680MiB / 15109MiB	96%
GPT1+2layer+AdamW	5410MiB / 15109MiB	98%
GPT1+4layer+AdamW	6680MiB / 15109MiB	86%

Table 2: GPU memory for each experiments

First thing to notice from the table 2 above, is that the first 9 configs have the same Memory-Usage, however the four last experiments used memory differently. The optimizer did not affect the memory usage, the 9 first experiments had 1 layer so the same number of parameters that is why the same memory usage.
In contrast, when we increase the number of layer the memory usage increase too. For example, LSTM+1layer+Adam had 6364MiB / 15109MiB memory usage in the other the same model with 4 layers had 6680MiB / 15109MiB.
However, we have example where the number of layers increased but the memory usage decreased: LSTM+1layer+AdamW 6364MiB / 15109MiB vs.LSTM+2layer+AdamW 5408MiB / 15109MiB. Here, we notice that GPU utilization decreased from 95% to 82%.

8. Comment on the overfitting behavior of the various models you trained, under different hyper-parameter settings. Did a particular class of models overfit more easily than the others? Can you make an informed guess of the various steps a practitioner can take to prevent overfitting in this case? (You might want to refer to sets of experiments 2, 9, 10 for the LSTM and 6, 11, 12 for GPT – that evaluate models of increasing capacities).

   LSTM and GPT1 overfit with particular settings. However LSTM overfit more easily than GPT1 this is because we can clearly see the difference between experiment 2 and experiments 9 and 10 where the gap between train PPL and validation PPL is larger.In the other hand, GPT1 the overfitting is clearly shown in the experiment 12.
   To prevent overfitting, we can train the model on more examples or train the models for longer time. Also, we can try regularization methods.