NLP Independent Project 3

Brandon Peck
bjp9pq

Note:
The number of experiments and perplexity is limited by the low number of epochs I was able run due to limited computing power. I suspect with more epochs the perplexities would be lower since the LSTM could relearn sequences from the data. I did, however, train the model in pytorch with cuda on a gpu.

2) Recurrent Neural Network Language Models

3)Simple RNN LM
Perplexity of training data: **868.40**
Perplexity of development data: **14699.01**

4) Stacked LSTM

Number of LSTM layers	Perplexity of training data	Perplexity of dev data
N = 1	994.93	18918.29
N = 2	1282.166	17488.85
N = 3	1329.27	17433.41

Best N: **3**
Perplexity of training data when n = 3: **1329.27**
Perplexity of development data when n = 3: **17433.41**
Intuition:
Although N=2 has a lower perplexity on the training data, when reporting these results, I'll consider development data to be a better indicator of the perplexity since the model has not seen those sentences and could be memorizing sequences in the training data. A stacked lstm allows for a more complex representation of the data and would be useful if there is a large dataset with diverse representations. While this dataset is not too large the model was able to learn more representations from the text with more layers. I empirically observed this when looking at the predicted words of the model when it had 1 and 3 lstm layers. The 3 layer lstm presented more diverse words. However, I suspect with more epochs and data the results could change, where the 2 layer lstm could better learn the word embedding dependencies and we might see the 3 layer overfit.

5) Optimization

Optimization	Perplexity of training data	Perplexity of dev data
SGD with Momentum	1174.77	19380.46
Adam	612.274	54765.47
AdaGrad	685.13	152135.03

Best model: **SGD with Momentum**
Best model perplexity on training: **1174.77**
Best model perplexity on development data: **19380.46**
Intuition:

SGD with momentum memorizes a previous step's update and uses that to modify and smooth the next update when learning. This gives it an opportunity to approach a minima of the loss quicker and more accurately with each time step, so if the momentum is chosen adequately it makes intuitive sense that this model would perform better than learning without momentum. AdaGrad adapts the learning rate based on the sparseness of parameters in the model. This performed the worst which is likely because the data may not be sparse, containing regular phrases and uses the same words consistently.

6) Model Size

(input dim, hidden dim)	Perplexity of training data	Perplexity of dev data
(32, 256)	1089.325	9337.14
(256, 256)	924.82	9722.88
(16, 256)	1192.24	9332.4

Input dimension of best model: **16**
Hidden dimension of best model: **256**
Best model perplexity on training: **1192.24**
Best model perplexity on development data: **9332.4**
Intuition:
I chose these combinations of input and hidden dimensions for the following reasons. Considering the the lstm hidden dimension is transformed to the output vocab size and the vocab size is larger than the hidden dimension, I wanted to see if increasing the hidden dimension size allowed for more information and word embedding associations to be kept and result in lower perplexity. In one experiment I keep the input dimension the same as in the simple rnn, 32, but increase the hidden dimension to 256. This allows me to see if more word associations are made by keeping larger hidden matrices. In the next experiment I try changing the input and hidden dimensions to much larger matrices. The hope here is that more relationships between words inputted and in previous time steps can be maintained in the lstm. The last experiment is to bottleneck the input and allow the data to be represented largely in the hidden dimension. Comparing this to the model with large input and hidden dimensions would show how much prediction depends on current input compared to previous hidden dimensions since if this resulted in a lower perplexity, which it did, it could be reasoned that previous time step data was more important for prediction. From the results I could analyze that current input is less relevant than previous hidden input with this dataset and the word embedding relationships are complex enough to warrant a higher hidden dimension.

*number of experiments were limited by computation resource

7)Min-batch

Batch Sizes	Perplexity of training data	Perplexity of dev data
16	5885.22	8423.05
24	8812.40	1127.42
32	8933.57	20844.24
64	Cuda out of memory	Cuda out of memory

To implement this section I had to pad the sequences with a null tag, pack them, run them through the model, then unpack them and calculate the loss without taking into account the padded values. I used and article which I linked at the bottom to help with some of the code.

Do mini-batch sizes make a difference? Yes
Best batch size: 16
Best model perplexity on training: 5885.22
Best model perplexity on development data: 8423.05
Intuition:
By adding more dimensions to the matrices in the lstm to train batches of sentences simultaneously the throughput of the system is increase but the accuracy is not necessarily. Since the lstm is learning from the batched data separately, it may not be exposed to the same data and so learn different representations of the embeddings. However if there are longer term dependencies in the data and the batch size moves those dependencies closer together by creating an offset for each batch layer in the lstm, the lstm may better pick these up. This will not likely benefit our simple project of word generation. It seems for this project a lower batch size is beneficial as the lstm can learn more word dependencies given the limited number of sentences. The batch size of 16 outperformed the rest likely because the lstm saw more data per epoch.

I referenced these sources when writing my code:
https://pytorch.org/tutorials/beginner/nlp/sequence_models_tutorial.html

https://towardsdatascience.com/taming-lstms-variable-sized-mini-batches-and-why-pytorch-is-good-for-your-health-61d35642972e