E1-246 Assignment 2

G P Shrivatsa Bhargav
SR 14865
bhargavs@iisc.ac.in

1 Tasks

1.1 Task1

Character level LSTM based model

1.2 Task 2

Token level LSTM based model

1.3 Task 3

Sentence generation

2 Dataset

The dataset used is the Gutenberg corpus. There are over 2 million words in the corpus. The documents in the corpus are written by 12 authors during different time periods. Hence the dataset has a wide range of writing styles.

3 Models

Character and Word level models are presented in this section

3.1 Character level neural models

3.1.1 Data

The Gutenberg corpus has over 11.8M characters. Due to time and computing power limitations, experiments are performed on two subsets of this corpus.

- Subset1: All works of Austen.
- Subset2: Works of Austen, Bryant, Burgess, Carroll and Chesterton.

Both the subsets are used without any preprocessing(i.e all characters are kept). The one-hot encoded characters are input to the models.

Subset	Train	Validate	Test	num chars
Subset1	1.9M	39k	41k	81
Subset2	3.6M	78k	71K	94

Table 1: Dataset sizes (number of characters)

2-Layer LSTM (C1)

- Data: Subset1
- Model: Input →Dropout(0.45) →LSTM(512) →Dropout(0.45) →LSTM(512) →Fully connected →Softmax →Categorical cross entropy loss.
- Optimizer: RMSProp. Learning rate=0.001, rho=0.9
- BPTT: 80
- Batch size: 128
- Number of parameters: 3.3M
- Perplexity: 3.18

3-Layer LSTM (C2)

- Data: Subset2
- Model: Input →LSTM(512) →Dropout(0.45) →LSTM(512) →Dropout(0.45) →LSTM(512) →Dropout(0.45) →Fully connected →Softmax →Categorical cross entropy loss.
- Optimizer: Adam. Learning rate=0.001. Remaining parameters are set as described in the paper(Kingma and Ba, 2014).
- BPTT: 100
- Batch size: 128

- Number of parameters: 5.4M
- Perplexity: 3.32

3.2 Word level neural models

3.2.1 Data

The entire Gutenberg corpus of 2M words is used to build these models. The data is converted to lower case and all punctuation except comma and periods were removed.
About half the dataset (Bible, Shakespeare) have a completely different writing style. The model would not generalize well if it were trained mainly on this portion of the data. So the data is shuffled in chunks of 50 sentences.The same shuffled data is used for all the experiments.
Some words in the corpus were marked as unknown, denoted by the special token <unk>to handle out of vocabulary words during test time. 95%, 2.5% and 2.5% of the corpus are used for training,validation and testing respectively. The following versions of the dataset are created, each differing in the way in which the words as marked as <unk>.

- Data1: Words not occurring in the pre-trained GloVe embeddings(300 dimensional, 2.2M vocab)(Pennington et al., 2014) are marked as <unk>. Vocab size = 33.7K. Number of <unk>instances in training data = 19.7K
- Data2: Words occurring fewer than 10 times are marked <unk>. Vocab size = 9.5K. Number of <unk>instances in training data = 87.6K

Training setting: The loss function used is a sparse categorical cross entropy. This saves us from building one-hot representations of the target words during training.
The batch size is set to 20. To solve the exploding gradients problem, gradients are clipped if their norm exceeds 0.25. The learning rate is reduced(by factor of 0.25) if the magnitude of drop in validation accuracy is below 0.0001. In the interest of time, the max number of epochs is capped at 40. Also,the training is stopped if the validation accuracy doesn't decrease by more than 0.001 in 6 epochs. The result from the best epoch are reported.

3.3 Models trained on Data1

3.3.1 Vanilla LSTM based

1, 2 and 3 layer LSTMs with dropouts before and after LSTM layers are trained.Adam with learning rate 0.01 (other parameters are set according to the paper (Kingma and Ba, 2014)) is used as the optimizer. The training sequence length or BPTT is set to 35.

1 layer LSTM (W1)
- dropout: 0.4
- parameters: 29M
- word embeddings: 300D GloVe
- perplexity: 201

2 layer LSTM (W2)
- dropout: 0.5
- parameters: 31M
- word embeddings: 300D GloVe
- perplexity: 180

3 layer LSTM (W3)
- dropout: 0.5
- parameters: 33M
- word embeddings: 300D GloVe
- perplexity: 179

2 layer LSTM and learnt embeddings (W4)
- dropout: 0.5
- parameters: 31M
- word embeddings: learnt 300D embeddings
- perplexity: 183

2 layer LSTM and increased BPTT (W5)
- dropout: 0.5
- parameters: 31M
- word embeddings: pre-trained 300D GloVe
- BPTT: 80
- perplexity: 155

3.3.2 Vanilla LSTM with parameter tying

Model(W6): Trainable word embeddings of size 1500, 2 layer LSTM with 1500 hidden nodes, dropout of 0.65 after each layer, BPTT = 35. The word embeddings are learnt during training. Additionally, the weights of the embedding layer and the output layer are tied. This model was run for 40 epochs. Of all the models trained, this one gave the *best test set perplexity of 59.5*.

3.4 Models trained on Data2

Model W2 was run on Subset2 to see the effect of the vocabulary size on the perplexity. As expected, the perplexity dropped to 137.1.

4 Sentence generation

A temperature based sampling algorithm is used to generate sentences. At each time step, the probability distribution over all words is computed. This distribution can be made more or less conservative by decreasing and increasing the temperature respectively.Text generated from conservative distributions tend to be more like the training data(less diversity). Unconservative distributions allow for more diverse text generation but at the cost of more mistakes(spelling or grammatical).

```
sample(preds, temperature=1.0):
{
preds=log(preds)/temperature
exp_preds=exp(preds)
preds=exp_preds/sum(exp_preds)
sample |vocabulary| words
return the most frequent word
}
```

4.1 Example sentences

4.1.1 Character level models

Text generated using model C2

Diversity: 0.2
the proper thing which had been so much as he was at all that he was to be made a little strong and some time to be a decided consideration of her father, and the servant was all the servants of the sort of consideration of his own acquaintance. ”The desire of the same side to the same time, and
Diversity: 0.5
happy behaviour to her than she had been to be done. In the world that he had done a great deal of the grave of the scarlet end of the other and the two houses of the last red hair of the room, but on the month was carefully like the great little rest of the chair and saying, for the words he was a
Diversity: 1.0
a doubt; and though her domestic outtrick, the little right minutes which happened, in its opinion, with a friend of her empty, and darkness had lived up to his dead place. He was perfectly distressed in her tand servant to collect them all, as they were lasted with the sides of some young men, and
Diversity: 1.2
returning yesterday as ever to his doing itith; and Mrs. Elton must be perfectly trudger, such a lightness, breath and application in which Mr. Woodhouse should ever see Miss Bates, know were in painful father. She acknowledged that they call thorouchly but with the judgment in ancient fellow; and

4.1.2 Word level models

Text generated using model W5

Diversity: 0.2
and it with in and and and and and and and and it with in and and and and and and and and and and and and him and and
Diversity: 0.5
and a come to as in and and and and for and a how it with for and it with well come a presence to and he into and it
Diversity: 1.0
they thou 5 behind . not soul of for saw my gates one she hold some invisible like thereof of sick beast were discourse i was ones . . harriet
Diversity: 1.2
satisfaction to and cried moses . moses sun a warm due to story change and everywhere a virtue was but 27 again all . s hated but agitated heed and

Text generated using model W6

Mr . Weston had been shut up in the highest assurance ; but Lady Russell had the sound to attend to Highbury to some of the Musgroves ; the first elderly party , with the other in a GENIUS attentively good . He ’ s quite dismembered .” Whether Loose or Fish is like , but a man or a third gentleman are after all . What am I beside you ? Now , the lazy gentleman looked for the little Jackal .”’ And so and the Shepherd said , with a continual persuasion , ” Spike does not

5 Analysis

Character level models are smaller(5M vs 31M parameters) and faster to train compared to word level models. This is mainly because the number of unique characters in a corpus is much smaller than the number of unique words so the embedding and output layers are a lot smaller when compared to word level models. Simple one-hot encoding of characters works well and sophisticated representations like CBOW are not required. Out of vocabulary words don't need to be handled explicitly if the character set is chosen properly. As shown above, the character level models have learnt spelling, punctuation, capitalization really well.The model rarely makes any mistakes in these areas. Word level dependencies are also captured to some extent. Character level models need much less data to generate the same quality of text compared to word level models.

Word level models are larger and data hungry. The main reason is the size of the vocabulary. Softmax computation is a bottleneck and so is the training of the fully connected layer before it. The probability mass is spread thin and the gradient (and updates) become small so the training becomes slow. In both character and word level models, it is observed that simply stacking mode and more layers doesn't result in a better model. The gains saturated after the number of layers crosses 2. This effect was more pronounced in word level models. Increasing the embedding size and the size of the LSTM hidden layer is more beneficial(as shown in model W6) than stacking layers.

Tying the weights of the embedding layer and the output layer reduced the size of the model as well as the perplexity by 10 points compared to the non-parameter tied model. Weight tying acts as a regularizer by reducing the degrees of freedom of the model so even though model W6 is larger, it doesn't overfit.

The mini-batch size also had a role to play. When the models W1 to W5 were trained with larger batch sizes (512-1024), the perplexities on the training data were low but on the test set, they were around 300. But reducing the mini-batch size to 20 reduced the test set perplexities. The smaller batch sizes reach "wider" local minima whereas the larger batch sizes will reach "sharper" minima. The "wider" minima will generalize better. This is demonstrated by (Keskar et al., 2016)

As the size of the vocabulary was brought down artificially by marking more infrequent words as <unk>, the perplexity decreases. But the model is only learning to predict <unk>. So even though the perplexity is lower, the language isn't modeled well because the quality of text generated is poorer.

Model W4 shows that learning the word embeddings instead of using pre-trained vectors doesn't help much in this task. This is because the GloVe(Pennington et al., 2014) vectors are also trained in a similar fashion. Maybe if the training corpus size is high enough, learnt embeddings would outperform the pretrained ones on this corpus.

Word level bigram Kneser Ney model from the previous assignment gives a perplexity of 157 on this dataset. The count based models are much faster and cheap to compute. On this corpus, the bigram model is better than the vanilla lstm based models (W1 to W3). But models with larger embeddings and hidden vector sizes (like model W6) easily outperform the bigram model.

6 Source code

The source code for the mentioned models can be found at `https://github.com/gpsbhargav/NeuralLanguageModels`

7 Software used

The models were built in Keras using TensorFlow backed. NLTK is used for tokenization and preprocessing. Model W6 is built in PyTorch

8 Acknowledgements

Inspiration for character level models was taken from Andrej Karpathy's blog post . The model W6 was taken from the PyTorch repository . The model W6 was replicated in Keras but due to its large size and Keras's slower LSTM/CuDNNLSTM, each epoc needed over 3 hours. Due to time constraints, the PyTorch model was tuned instead.

References

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. 2016. On large-batch training for deep learning: Generalization gap and sharp minima. *CoRR*, abs/1609.04836.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.