Writing a Kim Possible Scene Using an LSTM Language Model

Julie Maria Petersen (s164510), Lise Styve (s153748)
Technical University of Denmark, 02456 Deep Learning
December 4, 2019

Introduction

Language models have increased in importance as they are able to generate realistic, coherent and human-like pieces of text.
A popular model is the Recurrent Neural Network (RNN) with **Long Short-Term Memory (LSTM)** cells. It can be trained on large amounts of text from a chosen topic, and compute the conditional probability of a word, given the preceding sequence of words. In response to some initial words, it can generate similar text by itself.

Our goal is to build a language model that can generate a new Kim Possible scene.

Hyperparameters

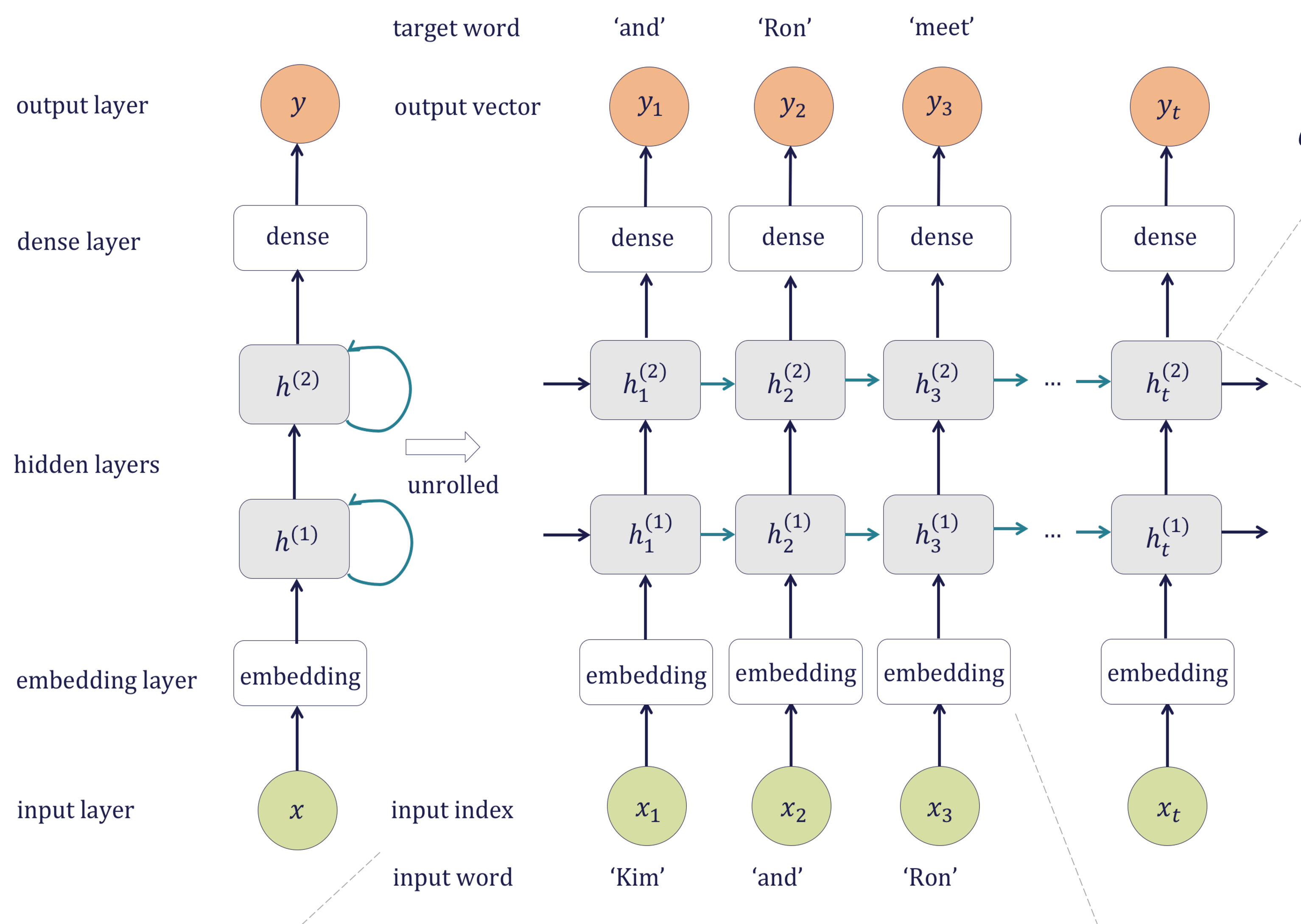
Model	Seq. size	Batch size	Embed. size	Hidden size	Layers	Epochs
Neural cache LSTM: Grave, Joulin, Usunier (2016)	30	20	1024	1024	1	50
Non-reg. LSTM: Zaremba, Sutskever, Vinyals (2015)	20	20	200	200	2	15
AWD-LSTM: Merity, Keskar, Socher (2017)	50	40	400	1150	3	750
LSTM: Our implementation	35	40	250	500	1	15

Regularization

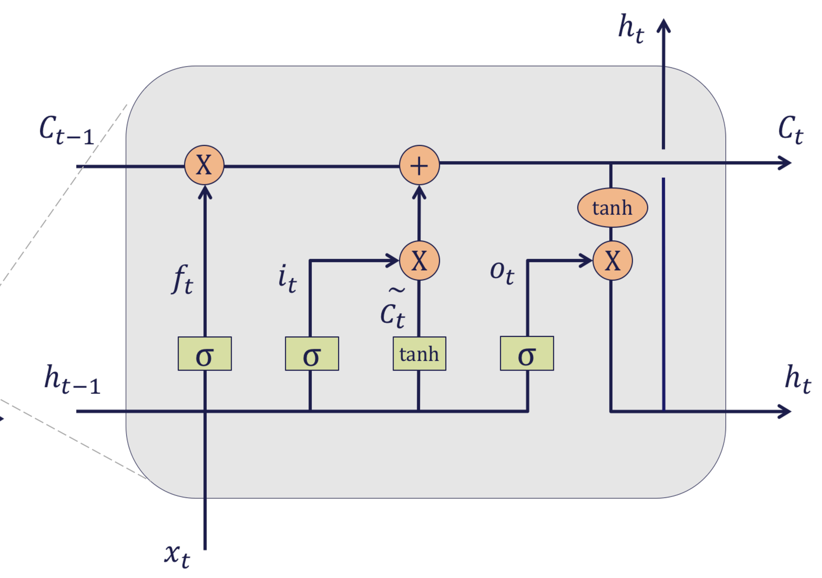
- L^2-regularization/weight decay: 2e-5
- Gradient clipping with max. norm: 5.0

Model

LSTM Neural Network



LSTM cell structure



$i_t = \sigma(W_i[h_{t-1}, x_t]+b_i)$ input gate

$f_t = \sigma(W_f[h_{t-1}, x_t]+b_f)$ forget gate

$o_t = \sigma(W_o[h_{t-1}, x_t]+b_o)$ output gate

$\tilde{c_t} = \tanh(W_C[h_{t-1}, x_t]+b_C)$ new memory cell

$C_t = f_t * C_{t-1} + i_t * \tilde{c_t}$ final memory cell

$h_t = o_t * \tanh(C_t)$

Word to index

Input words: 'Kim and Ron meet at Bueno Nacho.'

Tokenizing: ['Kim', 'and', 'Ron', 'meet', 'at', 'bueno', nacho','.']

Indexing: {'Kim': 5, 'and': 3, 'Ron': 7, 'meet': 445, 'at': 23, 'bueno': 178, 'nacho': 290, '.': 1}

Input indices: [5, 7, 445, 23, 178, 290, 1]

Embedding

'Kim': 5 $\longrightarrow \begin{pmatrix} 0.67 \\ 0.50 \\ -0.68 \\ 0.46 \\ -1.18 \\ \dots \\ 0.42 \end{pmatrix}$ 'Ron': 7 $\longrightarrow \begin{pmatrix} -1.12 \\ 0.31 \\ 0.54 \\ -0.02 \\ -0.98 \\ \dots \\ 1.13 \end{pmatrix}$

Computation

```
class RNNModule(nn.Module):

    def __init__(self, n_vocab, seq_size,
                 embedding_size, lstm_size, num_layers):
        super(RNNModule, self).__init__()
        self.seq_size = seq_size
        self.lstm_size = lstm_size
        self.num_layers = num_layers

        self.embedding = nn.Embedding(n_vocab, embedding_size)
        self.lstm = nn.LSTM(embedding_size, lstm_size,
                            num_layers, batch_first=True)
        self.dense = nn.Linear(lstm_size, n_vocab)

    def forward(self, x, prev_state):
        embed = self.embedding(x)
        output, state = self.lstm(embed, prev_state)
        output = self.dense(output)
        return output, state

    def zero_state(self, batch_size):
        return (torch.zeros(self.num_layers, batch_size,
                            self.lstm_size),
                torch.zeros(self.num_layers, batch_size,
                            self.lstm_size))

def get_loss_and_train_op(net, lr, weight_decay):
    criterion = nn.CrossEntropyLoss()
    optimizer = torch.optim.Adam(net.parameters(),
                                 lr, weight_decay)
    return criterion, optimizer
```

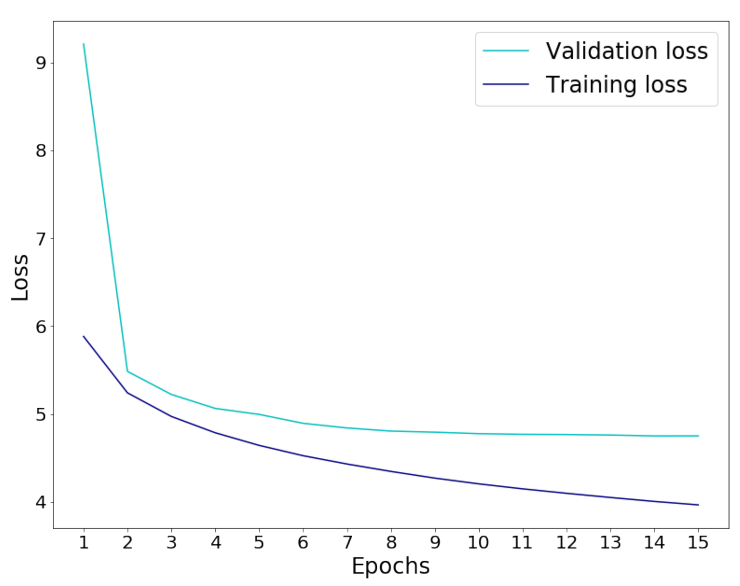
Conclusions

An LSTM Neural Network was successfully implemented. Different sets of hyperparameters were trained using regularization methods L^2 **weight decay** and **gradient clipping**. For the prediction, the decoding algorithm **top k sampling** has been used. The performance of the network will be sought optimized by further introducing some of the following **regularization methods**:

- Drop connect
- Variable sequence length
- Weight tying
- Independent embedding size and hidden size

Performance

Model	Validation perplexity	Test perplexity
Neural cache LSTM: Grave, Joulin, Usunier (2016)	86.9	72.1
Non-reg. LSTM: Zaremba, Sutskever, Vinyals (2015)	120.7	114.5
AWD-LSTM: Merity, Keskar, Socher (2017)	60.0	57.3
LSTM: Our implementation	113.9	112.4



Text generation

'Wade says he had been following the kimmunicator'

'Rufus is still obsessing over the world'

'He then says that she has to deal with her. She tells her to get the kimmunicator, and Ron is forced to get out the kimmunicator.'

' Kim then tells her mom that she is a sweetheart'

'As they leave Ron and Rufus are greatly enjoying the kimmunicator'

Decoding algorithm

- Top k sampling with fixed and adjusted k

References

[1] Aghajanyan, Armen. "Importance of Decoding Algorithms in Neural Text Generation." 19 May 2019, armenag.com/2019/05/16/importance-of-decoding-algorithms-in-neural-text-generation/.
[2] Grave, Edouard, et al. "Improving Neural Language Models with a Continuous Cache." ArXiv.org, 13 Dec. 2016, arxiv.org/abs/1612.04426.
[3] Karpathy, Andrej. "The Unreasonable Effectiveness of Recurrent Neural Networks." 21 May 2015, karpathy.github.io/2015/05/21/rnn-effectiveness/.
[4] Kim Possible Wiki. (2019). https://kimpossible.fandom.com/wiki/
[5] Merity, Stephen, et al. "Regularizing and Optimizing LSTM Language Models." 7 Aug. 2017, arxiv.org/abs/1708.02182.
[6] "Natural Language Processing with Deep Learning." CS 224N, Stanford University, 2019, web.stanford.edu/class/cs224n/index.html#schedule.
[7] Oinkina, and Hakyll. "Understanding LSTM Networks." Understanding LSTM Networks -- Colah's Blog, Colah's Blog, 27 Aug. 2015, colah.github.io/posts/2015-08-Understanding-LSTMs/.
[8] Wadsworth, Christina, and Raphael Palefsky-Smith. "Implementing A Neural Cache LSTM." Stanford University, 2017, web.stanford.edu/class/archive/cs/cs224n/cs224n.1174/reports/2760222.pdf.
[9] Zaremba, Wojciech, et al. "Recurrent Neural Network Regularization." ArXiv.org, 19 Feb. 2015, arxiv.org/abs/1409.2329.