Transfer Learning in Stacked LSTM's for Text Generation

Alexander Fischer
University of Massachusetts, Amherst
afischer@umass.edu

Abstract

Recurrent neural networks, like all machine learning models, require a large amount of training data in order to be effective. One way to mitigate this problem is with transfer learning. We propose performing transfer learning on stacked LSTM's using text data. We demonstrate the effectiveness of this technique by training a stacked LSTM on a large corpus of text data (we used books from project Gutenberg), then transferring the network weights to be trained on a very small corpus of text data(we used song lyrics from the Beatles). Our network learned from the transferred network outperformed the best LSTM we could train on only the small text corpus, establishing the viability of transfer learning in this fashion.

1. Introduction

Consider the task of training a recurrent neural network to generate text of a certain type, *e.g*. text matching the style of a particular author. This task requires a large amount of training data, so it is generally infeasible unless there exists a large corpus of such text. For example, one can train an RNN to output text similar to, say, Shakespeare's writings, as Shakespeare was a prolific writer and there exists a large corpus of his work, but one cannot train an effective RNN to output text similar to, say, lyrics of a specific musician, as one musician generally cannot write enough songs to properly train an RNN.

Humans have no problem understanding the style of some text given only a small sample of said text. Why can they learn so much faster than RNN's? Because RNN's have to learn the entire structure of the English language from their training data, whereas humans know the structure of the English language before thinking about any textual styles. Humans *transfer* their knowledge of the low-level structure of English—the spelling, grammar, et cetera—to the task of recognizing and generating text of a certain style, whereas RNN's do not.

We can improve the performance of RNN's on small training datasets by first training them on larger, more general datasets—say, the large amounts of English text from any source—then train the resulting RNN to mimic a specific source of text for which there exists little training data by only training a subset of the RNN's weights on the more specific data. This will allow the RNN to have a high enough model capacity to understand and generate text that follows the complex rules of the English language, while preventing it from overfitting the small amount of more specific training data.

This is analogous to transfer learning tasks in computer vision. It is common to the high-level features of a high-capacity neural network trained on a large dataset such as ImageNet as inputs to a small neural network trained on a smaller dataset [7] [10]. However, this is more more complicated with recurrent neural networks because it is not immediately obvious which activations correspond to lower-level features which will be invariant across different text sources (such as spelling, grammar, et cetera) and which activations correspond to higher-level features (such as word choice, subject matter, and tone), unlike with CNN's for computer vision in which the later layers correspond to higher-level features.

Previous work [6] has adapted a pretrained speech recognition RNN to work with a specific speaker by training a small subset of the RNN on the speaker's voice; however, to the best of our knowledge, this technique has not been attempted for text generation. Such work designed the RNN to have a specific subset of weights that is speaker dependent and thus will be trained post-tranfser.

1.1. Stacked LSTM's

We will use stacked long short term memory (LSTM) recurrent neural networks [3] [1] for our text-generation networks. Stacked LSTM's consist of several distinct LSTM layers composed with each other. At every timestep, the input data is fed into the first LSTM, then the output of that LSTM acts as the input of the next input LSTM, and more generally the output of each LSTM acts as the input of each successive LSTM. The output of the final LSTM is the output of the entire network at any given timestep.

We will perform transfer learning on stacked LSTM's by

training the whole network on a large dataset, then training a subset of the layers on a smaller dataset. To my knowledge, this training methodology has never been used with LSTM's.

2. Problem Statement

We have a small text dataset, and we want to train an LSTM to generate text based on that dataset; however, the size of the dataset limits the capacity of a model we can train before overfitting. Instead, we pretrain a higher-capacity stacked LSTM on a large text dataset that is similar to our goal text, then train a subset of the LSTM's weights on our goal text.

2.1. Datasets

We used three data sources for this project. The first, primary dataset is all song lyrics from the Beatles. This dataset is small compared to other datasets used to train recurrent neural networks. Combined, it contains 181 songs for a total of 146,000 characters. Since we used one third of the data each for training, validation, and test data, the training data for this dataset was tiny compared to most datasets used to train neural networks. This is the dataset on which we will focus our attempts to build an accurate network for text generation.

We used two secondary datasets that we used to pretrain networks which we then transferred to train on our primary dataset. The first secondary dataset was a collection [4] of lyrics from 55,000 songs from Kaggle. Its much larger size allowed us to train a much higher capacity network on it. This text is quite similar to Beatles lyrics, so we expected transfer learning to work quite well between this dataset and the Beatles dataset.

The second secondary dataset was a collection of books from Project Gutenberg [5], an organization that focuses on digitizing works of literature in the public domain. Because this dataset is so massive, we sampled several books from the dataset and pretrained a network on those. Like the Kaggle dataset, the size of this dataset allowed us to train a much higher capacity network on it. This dataset is only somewhat similar to Beatles lyrics, as the public-domain books are from several decades prior and feature very different text style, so we were unsure if transferring a network trained on this dataset to the Beatles task would work.

3. Technical Approach

Our first task was to train the best network we can on only the Beatles lyrics dataset. This network served as a baseline to see how much our transfer learning algorithms improve upon the default. For this task and all subsequent tasks, we used a stacked variant [1] of a vanilla LSTM [2] that processes text on a character level. Our character representation consisted of one-hot vectors of size 128, as we did not see any characters beyond the 128 characters in the first half of ASCII in any of our datasets (after appropriate preprocessing). Our LSTM's will had 64 hidden memory units in each layer, and a fully connected linear layer that maps $\mathbb{R}^{64} \rightarrow \mathbb{R}^{128}$ mapped from the top layer's hidden state to a vector of size 128 that represents class scores for the output character at each timestep. See figure Figure1 for a three layer example of our architecture.

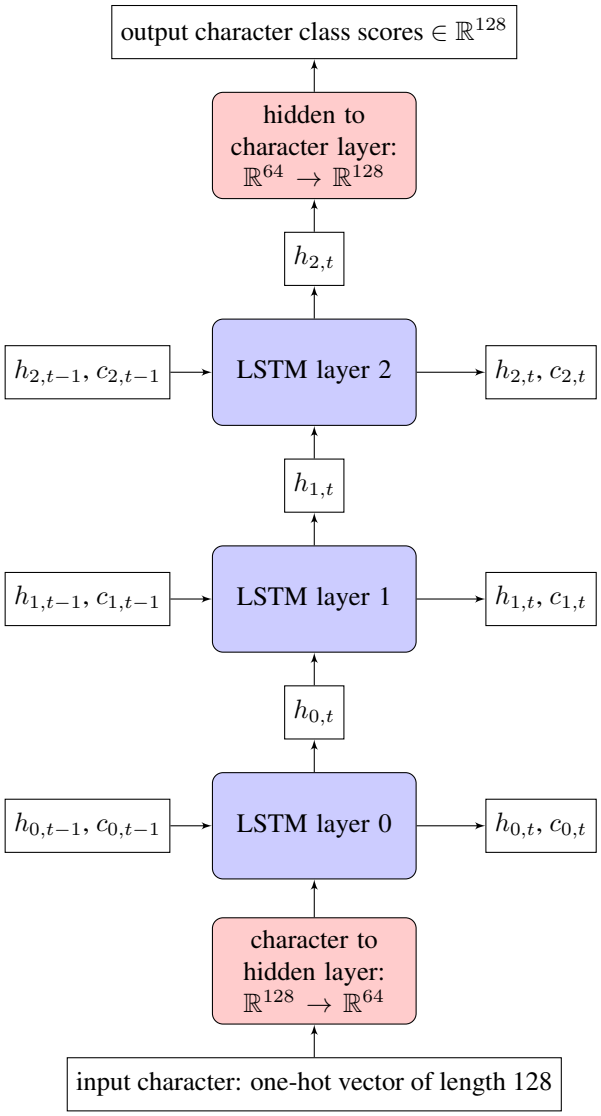

Figure 1. Our 3-layer stacked LSTM architecture. $h_{i,t}$ and $c_{i,t}$ refer to the hidden state and memory state, respectively, of level i of the stack at timestep t. Blue boxes represent individual LSTM layers, and red boxes represent fully connected linear layers.

Once we had a baseline network trained only on Beatles data, we began the process of transfer learning. This started with pretraining higher capacity networks on the large datasets we had, namely the song lyrics dataset and the

Project Gutenberg dataset. We used a three layer stacked LSTM architecture, as seen in Figure1. We used RMSProp [9] (our hyperparameter values were a learning rate of 10^{-2}, and $\alpha = 0.99$) to pretrain these networks, and we used the cross entropy loss function. Every epoch, we tested the network against the validation data, and at the end of training (when the validation loss ceased decreasing substantially or started increasing), we chose the model parameters that resulted in the minimum validation loss to be the parameters of our pretrained network for that dataset.

Once we had our pretrained networks, we turned to the task of training them on Beatles lyrics data. This consisted of training a subset of the layers of the full three layer pretrained models on the Beatles data (along with the final, hidden-state-to-character layer). We tried all nontrivial subsets (ie, not including $\emptyset$ and the subset equal to the original set) of the three layers (and retrained the hidden-state-to-character each time) with this approach. Similarly to our pretraining process, after each epoch of training with the Beatles data, we computed the loss on the validation set, and ended training after the validation loss ceased decreasing or started increasing. For each pretrained network and subset of layers that we trained, we used the model with the minimum validation loss as the model upon which we based our results.

4. Experimental Results

The best network we trained using only Beatles data was a 2-layer stacked LSTM with dropout [8] 0.2. and hidden layer size 64. This network achieved a minimum validation set loss of 1.039, and the test set loss on that network was 1.179. In general, we experienced much higher test set error than validation set error when working with the Beatles lyrics data. This is likely due to random variance in how hard the validation set and test set examples are to predict, as each set is quite small (60 songs) and can exhibit high variance in how hard its members are to predict.

Even with dropout, the best network we trained using only the Beatles data exhibited a large disparity between the training set error and validation/test set error. This is indicative of overfitting, which is an unsurprising result with such a small dataset.

As expected, networks that were learned starting from transferred weights trained on larger datasets performed better on the small Beatles dataset. The results of our experiments can be see in Table1 and Table2.

Predictably, networks pretrained on song lyrics data performed better than networks pretrained on Project Gutenberg data. Remarkably, networks pretrained on song lyrics data *without any retraining on Beatles data* exhibited lower validation set loss on Beatles data than networks only trained on the Beatles data, indicating how similar the two datasets are (although the pretrained, unretrained networks exhibited higher training set loss than networks only trained on the Beatles data). Networks pretrained on Project Gutenberg data without any retraining exhibited much higher validation set loss on the Beatles data than networks trained only Beatles data, which reinforces the notion that the Project Gutenberg data is less similar to our target Beatles data than the song lyrics data.

Layers retrained	**Minimum validation loss**	**Test loss on minimum validation loss network**
1, 2	0.962	1.100
0, 2	0.954	1.090
0, 1	0.964	1.101
0	1.001	1.133
1	0.983	1.123
2	0.979	1.116

Table 1. Results for transferring a 3-layer stacked LSTM network trained on Project Gutenberg books to song lyrics from The Beatles. We used cross-entropy as our loss function. The best network we could train on only the Beatles dataset resulted in a validation loss of 1.039 and a training loss of 1.179.

Layers retrained	**Minimum validation loss**	**Test loss on minimum validation loss network**
1, 2	0.817	0.949
0, 2	0.816	0.950
0, 1	0.818	0.946
0	0.816	0.949
1	0.816	0.950
2	0.813	0.946

Table 2. Results for transferring a 3-layer stacked LSTM network trained on general song lyrics to song lyrics from The Beatles. We used cross-entropy as our loss function. The best network we could train on only the Beatles dataset resulted in a validation loss of 1.039 and a training loss of 1.179.

When re-training networks pretrained on Project Gutenberg data, re-training two layers resulted in better performance on Beatles lyrics than re-training one layer, although the exact layer(s) that we re-trained mattered less than how many layers we re-trained.

Networks pretrained on song lyrics data generally performed equally well on Beatles lyrics regardless of which and how many layers we retrained. The fact that networks pretrained on Project Gutenberg data required retraining two layers to achieve their best performance, while networks pretrained on song lyrics data only required retraining one layer is expected, as it reflects the fact that the song lyrics data is more similar to our target Beatles data

than the Project Gutenberg data.

Interestingly, networks pretrained on song lyrics data performed better when re-trained with a smaller learning rate that what worked well otherwise. We used a learning rate of 10^{-3} when retraining these networks, as opposed to 10^{-2} with all others. This matches the usual recommendation when fine-tuning transferred convolutional neural networks to reduce the learning rate post-transfer.

4.1. Example Generated Songs

While the above quantitative results give a rigorous, mathematical description of the performance increases that resulted from our transfer learning methodology, samples from the best models from each category can give us an intuitive, qualitative understanding of the improvements from our methods.

To sample songs from our network, we used a softmax layer on top of the character scores output with a temperature of TODO as our character probability distribution. Because our character encoding includes some nonprintable characters (namely, those charaacters with decimal value less than or equal to 31), we keep sampling characters at each timestep until we sample a printable character. TODO

5. Further Work

While some of the results from our transfer learning experiments match what we would expect given our knowledge of transfer learning with convolutional neural networks for computer vision, some of the results did not match. The performance of a network learned from a pretrained network on the Beatles dataset generally depended the number of layers we re-trained, but not the specific layer that we re-trained. For example, re-training the last layer of a 3-layer LSTM pretrained on lyrics data performed approximately the same as re-training the first layer.

This is not the case with convolutional neural networks; it is well known that re-training the last layer(s) of deep convolutional networks results in better performance when doing transfer learning [7] [10] because later layers of convolutional networks correspond to higher-level features of the image. The fact that this result does not hold with our approach to transfer learning suggests that the conventional wisdom of later layers corresponding to higher-level features of the input data does not hold with stacked LSTM's. Further work could thus focus on determining exactly what weights in stacked LSTM's (or recurrent neural networks in general) correspond to higher-level features of text. Retraining these weights should result in superior performance with transferring pretrained recurrent neural networks.

Another approach could focus on designing recurrent neural network architectures that have a natural notion of which parts of the network correspond to higher-level features of the input. Such architectures would enable transfer learning with text data, and they may lead to more interpretable models, just as convolutional neural networks with their interpretable lower layers have led to a greater understanding of how neural networks process images.

6. Conclusion

We demonstrated the effectiveness of a novel approach to transfer learning with LSTM's for text generation. We used this technique to train a network that predicted and generated text on a small dataset (song lyrics from The Beatles) better than we could do by training a network only on said small dataset.

References

[1] A. Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

[2] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber. Lstm: A search space odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, 28(10):2222–2232, 2017.

[3] S. Hochreiter, Killough, and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[4] S. Kuznetsov. 55000+ song lyrics. `https://www.kaggle.com/mousehead/songlyrics/home`.

[5] S. Lahiri. Complexity of Word Collocation Networks: A Preliminary Structural Analysis. In *Proceedings of the Student Research Workshop at the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 96–105, Gothenburg, Sweden, April 2014. Association for Computational Linguistics.

[6] Y. Miao and F. Metze. On speaker adaptation of long short-term memory recurrent neural networks. In *Interspeech*, 2015.

[7] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. CNN features off-the-shelf: an astounding baseline for recognition. In *CVPR*, 2014.

[8] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[9] T. Tieleman and G. Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

[10] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? In *NIPS*, 2014.