# Transfer Learning in Recurrent Neural Networks 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 will try to find an algorithm to train a recurrent neural network for text generation to mimic specific writing styles for which there exists little training data by first training on more general text for which there exists a large amount of training data, then training a subset of the resulting recurrent neural network using the small amount of available training data for the more specific writing style.*

## 1. Introduction

Consider the task of training a recurrent neural (RNN) network to generate text of a certain type, *e.g*. text matching the style of a particular author or song lyrics. This task requires a large amount of training data, so it is generally infeasible unless there exists a large corpus of similar 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 [5] [6]. 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 authors (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 [4] has adapted a pre-trained 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-transer, suggesting that if we cannot discover a natural notion of weights that affect high-level text features in existing RNN architectures, we may have to design our own architecture that explicitly includes such weights.

Note that this project milestone is a different project idea that what I initially proposed.

## 2. Problem Statement

We wish to train an RNN to generate text that mimics some pre-existing text for which we have only a small corpus of, too small to fully train an RNN. We must train an RNN to generate English text using a large corpus of pre-existing English text, then further train the resulting RNN on the specific text to generate text in the desired style.

The principal challenge is to determine how to train a pre-trained RNN to mimic a certain textual style. There is no obvious notion of which activation in an RNN correspond to high-level features associated with textual style, and we will have to determine that in order to effectively train a limited subset of the RNN.

## 3. Technical Approach

We will use an LSTM [3] as our recurrent neural network for text generation. Our source of English text for pre-training the model will include books from project Gutenberg, modern newspaper articles, and movie dialogs [1].

To train the pre-trained network on the smaller, more specific text, we will train a subset of the weights of the pre-trained LSTM that produce activations that correspond to higher-level features of text. We will have to do some investigation in order to determine which weights to train. This investigation will take the form of feeding the pre-trained LSTM text with various low-level and high-level features modified in specific ways and seeing which memory cells are correlated with various statistical features of the input text.

For example, we will feed the pre-trained LSTM text with a variety of sentiments, from happy to sad, and see which memory units and hidden-layer activations have the highest correlation with the sentiment of the input text. These memory units and hidden layer activations should correspond to high-level features, meaning the weights that affect them should be trained on the smaller, more specific corpus.

We will also feed the pre-trained LSTM text, then subsequently that same text but with low-level features of the text changed, such as spelling errors and slightly different wording. We will see which memory units and hidden layer activations change in response to these changes, and which ones do not change. These memory units and hidden layer activations should correspond to low-level features, meaning the weights that affect them should be held constant after pre-training.

Ideally, these procedures should yield an intuitive heuristic as to which weights affect high-level and low-level features of text, rather than a hacked-together list of weights which seem to do what we want. For example, the later layers in convolution neural networks for computer vision correspond to higher-level features; we would like to discover a similar guiding principle for LSTM's. Such a guiding principle would allow us to train a much more principled, complete set of weights on our smaller, more specific corpus. It would also allow easier generalization of our results to different RNN architectures and to problems beyond English text, rather than having to repeat the aforementioned weight investigation procedures every time we wish to try a different pre-trained network.

To evaluate this approach, we will train a pre-trained LSTM on smaller and smaller text datasets and see how small the dataset can be before the LSTM overfits it. This should be much smaller than the smallest datasets that an LSTM can be trained on from scratch without overfitting.

We will use both objective and subjective metrics to measure the performance of our LSTM's. We will use the standard test-set accuracy of our LSTM's trained from pre-trained complex LSTM's and compare them to the test-set accuracy of LSTM's trained from scratch on our training data. Also, for a subjective performance metric, we will subjectively compare the text generated by our LSTM's and trained-from-scratch LSTM's. It should be obvious upon reading both which network better learns the structure of the small training data by seeing which generated text seems to be a better generalization of the training data.

## 4. Intermediate/Preliminary Results

I have started gathering text corpora for pre-training an LSTM. I have also read a survey article on LSTM architectures [2] to determine what architectures to use for my experiments.

## References


[1] C. Danescu-Niculescu-Mizil and L. Lee. Chameleons in imagined conversations: A new approach to understanding coordination of linguistic style in dialogs. In *Proceedings of the Workshop on Cognitive Modeling and Computational Linguistics, ACL 2011*, 2011.

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

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

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