Style-imitative Text Generation using LSTM-RNNs

Justina (Hyunji) Cho, Madhav Datt, and **Andrew Zhou**
Harvard University
{hjcho@g, madhav_datt@college, andrewzhou@college}.harvard.edu

Abstract

We develop a system using long short-term memory recurrent neural networks to generate style-imitative text based on *Sherlock Holmes* novels. We investigate the performance of this system against that of Markov models, and vary hyperparameters (specifically, node and hidden layer numbers as well as training epochs) for LSTM-RNNs to gauge the effectiveness of particular system configurations in generating text, and evaluate results using a modification to the BLEU metric.

1 Introduction

The problem of generating text in a particular style may be naïvely approached using a Markov model to learn the probability of the $n + 1$st character being any character c given the first n characters. This model, however, is faced with many limitations: the Markov assumption—that the current state is dependent solely on the past state—fails to model long-term dependencies between far-apart characters, and the state space explodes as n increases ([13]).

Recurrent neural networks are a powerful tool for text generation as they may model long-term dependencies and non-linear relations between text features. Moreover, the LSTM network architecture ([7]) provides a particularly effective means of improving RNN training by hastening error backpropagation, and the Adam optimization algorithm ([9]) has proven efficient in optimizing the loss function by which error is measured.

The tasks of model selection and tuning hyperparameters speak to a more fundamental challenge: adapting a given system to best suit the problem it is meant to solve. With this challenge in mind, we investigate the performance LSTM-RNNs compared to that of Markov models, and design and employ an evaluation function, a modified version of the BLEU metric ([10]), to gauge quality of generated text for both.

To conclude, we synthesize our above work to provide a general framework for tuning the parameters of LSTM-RNNs to best suit the structure of any particular style-imitative text generation problem. Our results provide insight into the crucial question of model selection and refinement in artificial intelligence.

2 Background and Related Work

The mathematical underpinnings behind the Markov model as a means of text generation were laid out by Shannon in his seminal paper "A Mathematical Theory of Communication" ([12]). Jelinek ([8]) examines the limitations of this model, arguing that as the number of characters represented in each state n increases: 1) the probability tables will become increasingly sparse as the number of potential states increases drastically and thus will require infeasible amounts of data to populate, and b) the tables themselves would require infeasible amounts of memory to store. See Section 4.1 for a mathematical discussion of the Markov model.

Neural networks with backpropagation are generally the preferred approach in this domain and others for their capacity to suss out nonlinear relations in data. Rumelhart, Hinton, and Williams ([11]) describe the basic structure of a neural network: a layer of input units (in our case, a sequence of usually 100 characters); a number of intermediate hidden layers; and a layer of output units (in our case, a one-hot encoded array of length equal to the number of characters in the training set). Inputs are fed into the network, results are computed, and a result is generated, which is compared to the expected result; the network is modified to minimize the error. Backpropagation is a particular type of error-minimizing technique where gradient-based calculations are used to modify weights from output layer back toward the input layer; Rumelhart, Hinton, and Williams show backpropagation to be particularly effective. See Section 4.2 for a mathematical discussion of neural nets.

Recurrent neural networks have been shown by Graves to be excellent for text generation due to their capacity to learn from sequential/temporal datasets ([6]). RNNs are notable in that they possess loops: the data from one input time step are fed into the next time step, influencing the development of the network and allowing the network to model dependencies between time steps, and thus dependencies between distant characters. Graves specifically discusses the LSTM-RNN architecture, developed by Hochreiter and Schmidhuber ([7]), which employs a particular gradient-based method to backpropagate errors consistently rather than have them blow up or vanish. Adam ([9]) is a specific method for gradient-based neural network optimization that has been shown to be effective.

With relation to the above, we specifically design a Markov model text generator and employ the Keras deep learning library with Theano [1] backend and Adam optimizer to construct an LSTM-RNN for text generation.

3 Problem Specification

Our overall problem was to develop a procedure for hyperparameter tuning in LSTM-RNNs: a method for finding the best LSTM-RNN configuration for a particular training set and problem. Parameters included the number of nodes, hidden layers, and epochs run. The system is extensible to various other parameters as well; see the discussion of future work in Section 7. With this goal in mind, we carried out the following steps:

We developed both a Markov model-based system and an LSTM-RNN for generating text from a given corpus. We then designed a metric, based on the existing BLEU metric ([10]), to evaluate the quality of generated text by both coherence and overfitting to the training set. We then generated text using both systems, varying hyperparameters for each, and evaluated this text using our metric. We plotted the progression of text quality for any particular parameter configuration as the number of epochs increased, and analyzed our plots to arrive at conclusions concerning the fitness of a configuration for modeling the style of the training set. We genericized this process to produce a framework for LSTM-RNN hyperparameter tuning.

4 Theory and Algorithms

4.1 Markov Model

We model our system using a Markov chain where we learn transition probabilities $P(x_i|x_{i-(n-1)}, ..., x_{i-1})$, obtained from character occurrences in the Sherlock Holmes texts. Using an n-gram model, we predict the letter x_i, based on the letters $x_{i-(n-1)}, ...x_{i-1}$.

We incorporate smoothing to account for n-grams not observed in our training text. We use Laplace smoothing to assign a count of k to unseen n-grams, and add k to all observed n-grams. When predicting the next character in a sequence, we sample from the probability distribution based on the previous $n-1$ characters.

$$P_{LAP,k}(x) = \frac{c(x)+k}{N+k|X|}$$

4.2 Neural Networks

As mentioned in Section 2, Rumelhart, Hinton, and Williams ([11]) describe the neural network model in depth. The input layer of nodes x_i is fed an input: an array of real numbers, one for each node. Each of these inputs is weighted and passed to one or more nodes in the hidden layer of nodes y_j. These nodes y_j apply an activation function u_j to their weighted inputs, and produce an output themselves, which is weighted and passed to one or more nodes in the output layer of nodes z_k. Each output node applies its own activation function u'_k to its weighted inputs to produce the output array.

Given x_i is the input to node x_i, w_{ij} is the weight between x_i and y_j, and y_j is the output of node y_j, [11] gives the weighting of the inputs as:

$$in_j = \sum_i x_i w_{ij}$$

$$y_j = \frac{1}{1+e^{in_j}}$$

Backpropagation uses an error function E and changes each weight by $\delta w = \epsilon \delta E / \delta w$ for some ϵ, starting from the output layer and moving toward the input layer. We do not replicate the math; see [11].

4.3 RNNs and LSTM-RNNs

As mentioned in Section 2, LSTMs are excellent at capturing long-term dependencies [7] for sequential data because each time step takes as input the output of the previous time step. The LSTM model introduces a memory cell [5]. Each such memory cell is made up of four key components: input gate, a connection to itself, a forget gate [4] and an output gate. The input gate allows or blocks incoming signals from making changes to the current state. Whereas, the self-recurrent connection has a weight of 1 to ensure that the state of a memory cell can remain constant from one timestep to another. We use Keras' [3] LSTM implementation with a Theano backend [1].

At every step in time t, an entire layer of memory cells is updated. The Keras library computes the values for i_t, input gate, and $\bar{C}_t$, where $\bar{C}_t$ is the candidate value for states of the memory cells at t based on the explanation here, and as follows:

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i)$$

$$\bar{C}_t = tanh(W_c x_t + U_c h_{t-1} + b_c)$$

f_t, the activation of the memory cells forget gates [4] at time t :

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f)$$

Based on the value of the input gate activation i_t, the forget gate activation f_t and $\bar{C}_t$ value, the new state of memory cells, C_t is computer:

$$C_t = i_t \times \bar{C}_t + f_t \times C_{t-1}$$

$$\implies output_t = \sigma(W_o x_t + U_o h_{t-1} + V_o C_t + b_o)$$

$$\implies h_t = output_t \times tanh(C_t)$$

where, x_t is the input signal to the memory cell layer at time t, W_n, U_m and $V_k \quad \forall n, m, k$ are weight matrices, and b_i, b_f, b_c and b_o are bias vectors implemented by the Keras model as shown here.

4.4 Modified BLEU Metric

The BLEU evaluation method, along with other automatic evaluation metrics, have been shown to have a high correlation with human judgment for evaluation of machine generated text ([2]). We design a modification to BLEU ([10]), a method for automatic evaluation of machine translation, and re-purpose it to evaluate the quality and degree of over-fitting in text generated by our system.

We calculate *modified n-gram precision* using this NLTK toolkit as follows: all candidate n-gram counts and their corresponding maximum reference counts are collected. Candidate counts are clipped by their corresponding reference maximum value, summed, and divided by the total number of candidate n-grams. We define *evaluation sentence, ES* as the sentence/string of characters generated by our system, for evaluation. *Reference sentence, RS* refers to a single string with of the same length as the *ES*, and are generated by dividing a *reference corpus, RC* into a set of such reference strings.

$$\text{Modified n-gram Precision,} \quad p_n = \frac{\sum_{ngram \in ES} C_{clip}(ngram)}{\sum_{RS \in RC} \sum_{ngram' \in RS} C(ngram')}$$

where, clipped count for ngram, $C_{clip}(ngram) = \min(C(ngram), \max_{RS} C_{RS}(ngram))$, and $C(ngram) =$ ngram count in source.

Thus, we calculate a modified version of the original BLEU score ($BLEU'$) as follows:

$$BLEU'(n_1, N) = e^{\sum_{n=n_1}^{N} w_n \log(p_n)}, w_n = \text{weight of } n^{th} \text{ order n-gram}$$

We evaluate the quality of the generated text as $BLEU'(1,4)$, and use the presence of higher order n-grams in generated text to imply over-fitting of training data. We calculate overfitting as $BLEU'(5,8)$. All order n-grams are weighted equally with an arbitrary weighting constant $w_n = 0.25 \forall n \in [1,8]$.

5 Methods

5.1 Project Pipeline

See Figure 1 for a graphic that explains the pipeline of our project. We begin with a corpus of training data; we used `training/4-mod.txt` for the LSTM and `training/sherlock-all-mod.txt` for our Markov model. We then preprocessed the text with `training/preprocess.py`. We trained the LSTM on the Odyssey cluster and constructed the Markov model on our local computers; we trained multiple times with varied parameters: number of layers, number of nodes, and number of epochs for the LSTM and n-gram size for the Markov model. We then generated text particular to certain sets of parameters, calculated BLEU scores for each generated text using `evaluation/evaluation.py`, and plotted the results with `evaluation/plot.py`.

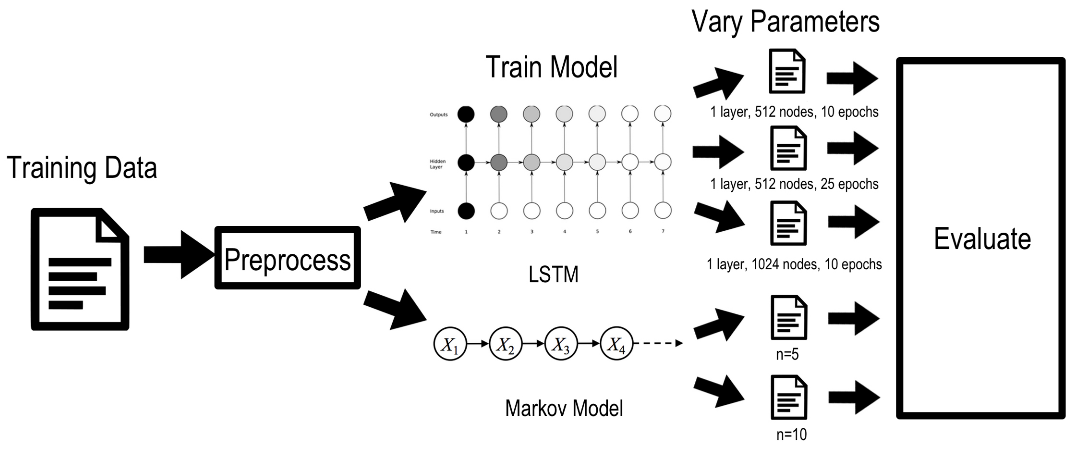

Training Data
Preprocess
Train Model
LSTM
Markov Model
Vary Parameters
1 layer, 512 nodes, 10 epochs
1 layer, 512 nodes, 25 epochs
1 layer, 1024 nodes, 10 epochs
n=5
n=10
Evaluate

Figure 1: Our general project pipeline

5.2 Text Acquisition

We acquired our corpora from Project Gutenberg. The file `training/4-mod.txt`, which we used as our principal corpus, comprises the first 100,000 characters of *The Sign of the Four*; the full text novel may be found here. Additionally, we use the preprocessed first 100 characters of `training/1.txt`, "to sherlock holmes she is always the woman. i have seldom heard him mention her under any other name" as a seed for text generation. We additionally have included the raw text of the remainder of the stories and novels in separate files and combined in `training/sherlock-all.txt` and the file `training/sherlock-all-mod.txt` is this full set of text processed. Our LSTM-RNN scripts may be run on any of the individual files in reasonable amounts of time (hours) on Odyssey using GPUs; our Markov model may be run on any individual file or the full text.

5.3 Preprocessing

In order to cut down on the number of characters used—and thus the size of the output layer of the LSTM-RNN—we have preprocessed our text with `training/preprocess.py`. We simply replace all uppercase letters with lowercase ones, and replace newlines with spaces. We do the latter as we do not concern ourselves with paragraph separations, choosing instead to focus on the sentence level.

5.4 Markov Model

We developed a Markov model for generating text, seen in `src/markov.py`. See Section A.1 for an explanation of how to generate text with the Markov model, Section 6.1 for generated text from the Markov model, and Section 6.3 for analysis.

5.5 LSTM-RNN

We developed a LSTM-RNN for generating text. Training is achieved by `src/train-generic.py`, generating text is achieved by `src\generate-generic.py`; both are run using scripts on the Odyssey Cluster. See Section A.2 for an explanation of how to train the LSTM-RNN and generate text with it, Section 6.2 for generated text from the LSTM-RNN, and Section 6.3 for analysis.

5.6 Odyssey Cluster

We trained our LSTM-RNN and generated results on Harvard's Odyssey computing cluster using GPUs. See Section A.2.1 for a replicable walkthrough of how we did so.

5.7 Text Evaluation

We developed a metric, used in `evaluation/evaluation.py` based on the existing BLEU metric ([10]) to evaluate the quality, in terms of both coherence and overfitting, of generated text. See Section 4.4 for an explanation of the metric.

5.8 Parameter Tuning

We first plotted the quality trends of various LSTM-RNN configurations versus the number of epochs run using `evaluation/plot.py`, and then tuned the various parameters of the models—nodes, layers, and epochs for LSTM-RNNs—to see the diverging behavior given different parameters. From these results, we develop a schema for hyperparameter tuning for LSTM-RNNs.

6 Experiments

6.1 Markov Model

We present various texts generated with our Markov model trained on `training/sherlock-all-mod.txt` for different n and smoothing s. We present also the overfit scores associated with each generated excerpt. The truncated versions of the generated texts have been presented and complete texts may be found in Appendix C.1

All texts were seeded with the last n characters of the preprocessed first sentence of "A Scandal in Bohemia" (`training/1.txt`):

```
to sherlock holmes she is always the woman.
```

$n = 1$, $smooth = 0.0001$;
based on previous 1 character, bigram model
overfit = 0, Full result: C.1

```
oo athay–mathavennit. i rprou weson amoundome sen thafot ms oweshen wig an thaien tho haliner tisitolobres t erine hem d, wardr, hed atinisupr glogrenn hin s theellibecerlsedishereas pp
```

$n = 3$, $smooth = 0.0001$;
based on previous 3 character, 4-gram model
overfit = 0, Full result: C.1

```
whited it was jacked hom the how to that therm one ster my and pare in his to my had overst. and totances had be more ver-capable serve to our posalue at at would, an a lith and proble, and
```

$n = 5$, $smooth = 0.0001$;
based on previous 5 characters, 6-gram model
overfit = 0, Full result: C.1

```
"i am may expensive his pipe to the may proof upon which occurred down than and suggesting finger-glasses over that we put your evidence that a fee, and his. hayes having clacking tracken
```

$n = 14$, $smooth = 0.0001$;
based on previous 14 characters, 15-gram model
overfit = 0.051, Full result: C.1

```
i have seldom heard him mention her under any other name. in his eyes she eclipses and predominates to so marked an extent that i could hardly have guessed, major john sholto, once of
```

We see here the typical behavior of a Markov model with different n and smoothing. First, we observe the effect of increasing the order of the Markov chain using n-grams, and a low smoothing constant. The smoothing discounts the probability of observed n-gram in order to assign probabilities to unseen n-grams. This allows us to deal with the zero-probability problem of unseen events. In this initial experiment, we choose a very low smoothing constant of 0.0001, thereby making it less likely to sample an unseen next character when generating text.

With $n = 1$, we are predicting the next character in the sequence based on the previous character. In this bigram model, we do not get coherent words because predicting the second character based on one previous is an insufficient size window to predict word-level relations. With $n = 3$, there is an improvement in the sequence of characters, and we begin generating coherent words. However, there is little sense and grammatical logic between words ("how looked told his"), and there are still sequences of characters which do not make up words ("utions ree jewellow"). With $n = 5$, we begin to bridge the gap between words for some semblance of grammatical sense ("but this wife! she"). We notice that $n = 1, 3, 5$ receive overfitting scores of 0 because the existing $n-$grams within the text are not large enough for our metric to begin penalizing them; however, $n = 14$ begins to demonstrate some measure of overfit.

As expected, when we increase the order of the Markov chain, we see improvements in generating words from our model because we can predict the next character based on the previous n characters. However, it is evident that the Markov chain does not remember long-term dependencies so coherence is limited to the order of the Markov chain. Though the text does appear fairly coherent for $n = 4, 7$, we notice the dual problems of inability to capture long-term relations and overfit. With small n, the Markov model can only capture relations within a small window; as n gets larger that window increases but overfit also increases, since the training set data for any particular n-gram for large n grows increasingly sparse.

We also observe the effect of smoothing in our model. When we add a constant k to all possible events, we assign a probability to novel unseen events. However, with an increasing value of k, novel events become more probable. This is observed when comparing $n = 5, smooth = 0.001$ and $n = 5, smooth = 0.0001$. With a larger n, the probability of observing a n-gram is relatively sparse, so with the addition of a higher value for a smoothing constant (0.001 vs. 0.0001), we give too much weight to novel events. Consequently, we then sample next characters with higher probability, and this results in generated text filled with random characters ($n = 5, smooth = 0.001$). In order to prevent this problem, we give even less weight to unseen events; however, this causes the generated text to become increasingly similar to the corpus.

The biggest problem is the explosion of time and memory requirements as n increases. We require time and memory exponential in n, and this requirement is impracticable for any model wishing to encode long-term relations within text. Both this problem and the overfitting problem with greater n render the Markov model impracticable, despite its reasonable coherence on the word-level.

6.2 LSTM-RNN

We present various texts generated with a 1 hidden layer, 512 node LSTM-RNN trained on `training/4-mod.txt` with window of 100 characters for different numbers of epochs and analyze the evolution of these texts. The truncated versions of the generated texts have been presented and complete texts may be found in Appendix C.2. We present also the coherence and overfit scores associated with each generated excerpt. The epoch numbers are not round because the model saves weights only when the epoch has improved upon the previous epoch's loss.

All texts were seeded with the preprocessed first 100 characters of "A Scandal in Bohemia" (`training/1.txt`):

to sherlock holmes she is always the woman. i have seldom heard him mention her under any other name

10 epochs; 0.08 coherence; 0 overfit: (Full result: C.2)

and the case to the tate of the caree to the cane tf the toee of the care th the toeee of the caree to the cane to the toees of the caree to the cane to the toees of the caree to the cane to the

100 epochs; 0.38 coherence; 0 overfit: (Full result: C.2)

ti the tine wf thaul make a marerl bistimrnt th thaaled et a phpein-whrnin! and i wishdyt soat her wan the kaits oe uuetpten aased oimtlonger and a were bertinned mome tpon the snpm, in the

251 epochs; 0.50 coherence; 0.42 overfit: (Full result: C.2)

tf hurpe shpugd bo alstinn a pairer whs. ant toon hw theses." "a dander ir thes wouk frien, seid he. "sheo is a cress buery tone " shi wisl anttan tonok has hht shdn in the herpe. kenk it the

503 epochs; 0.55 coherence; 0.47 overfit: (Full result: C.2)

an hhir. "i sise whit tou wel haclul, ant i have donn to myrsele to sou thit ael? an in iand sat toe out i'stond, and the nhr teien tp tia finsley of the ticket with a pin-point upon the inside of the

From the above excepts and Figure 2, which plots quality scores versus epochs for the 512 node, 1 hidden layer LSTM-RNN. We notice extreme repetition and incoherence for 10 epochs, improved coherence for 100 epochs, and from then on coherence hovers between 0.3 and 0.6 with further training. Moreover, as might be expected we see increasing overfitting with increasing coherence.

Looking at the generated texts themselves, we note that with 10 epochs, the LSTM-RNN is producing the same text repeatedly, a reasonable result given that little training has been done. The repetition of "to" and "the" is telling; the network has trained itself to replicate some of the most common words in the corpus.

With 100 epochs, we have mostly nonsense words due to underfitting, but these words begin to approximate words in the corpus and some sentence structure is present, including apparent quotations and common constructions such as "i," "the", "in," "a," and "to." There is no overfit here as few *n*-grams from the text are present in this excerpt.

With 251 and 503 epochs, we have increasing coherence as well as increasing overfit. The text frequently finds its way into long verbatim exerpts from the corpus, and the overfit metric punishes severely for these excerpts. These results are not acceptable generated texts due to massive overfit; we see that 100 epochs is the most appropriate weight set for this particular configuration.

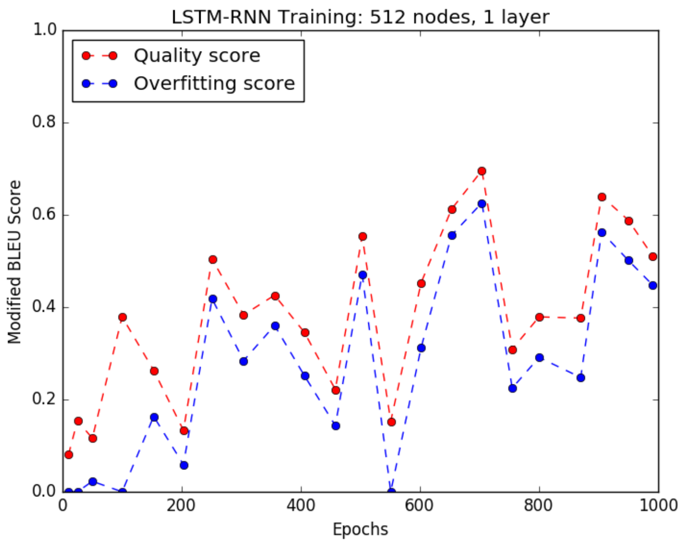


Figure 2: Quality scores versus epochs for 512 nodes and 1 hidden layer

6.3 Results

We present Figures 3 and 4, the analogues of Figure 6.3 for different hyperparameter configurations. In all these cases we see the progression of text quality as the number of training epochs increases. We see that for 128 nodes with 1 layer as well as 256 nodes with one layer (Figure 3), the coherence of the text does not measurably increase with increasing epochs, nor is overfitting significant at all. This trend indicates that the sophistication of the model is insufficiently complex to represent the structure of the problem; for this particular corpus (`training/4-mod.txt`) these models are significantly underfit.

We next observe the left-hand graph of Figure 4 in comparision to Figure 2. We see that the 512 node 1 layer model is significantly overfitting the training set as epochs approach 1000, but nonetheless does not reach coherence and overfit scores as high as the 1024 node 1 layer model at any point. This result is reasonable considering the relative sophistications of the two models: while 512 nodes can only ever moderately overfit the training set, 1024 nodes can overfit to an extreme degree with increasing epochs.

We observe the right-hand graph of Figure 4 to consider how the addition of a second hidden layer complicates the equation. Notably, at 303 epochs there is the following text:

303 epochs; 0.42 coherence; 0.25 overfit (Full text at Appendix C.3):

```
t. but there is no light from within that it was the baaurted to my feal. she othgr me siles as which
wou have been eodowed from the remiraoe of the treasure-seekers. the oes were alotered ...
```

Though moderately overfit at its close, the beginning exhibits regular sentence structure and approximate vocabulary. Though the 2 layer model goes on to radically overfit the training set—likely due to its sophistication—there is a period from 100-350 epochs with moderate $(0.2 - 0.4)$ coherence and low (< 0.2) overfit. It is likely the model has reached a "happy medium" between under and overfit here, and so this area should be looked to as a section of potential interest.

Overall, the one layer configurations exhibited a progression from extreme underfit to extreme overfit, with sections of the 512 node model exhibiting the most coherence. The 2-layer model, on the other hand, demonstrated an area of relative coherence with low overfit—perhaps a reflecting of the increased sophistication of the model—but later went on to overfit the model radically. Of the analyzed models, our suggestion would be to use the 512 node 1 layer system for 100 epochs, or the 2 layer system for between 100-350 epochs.

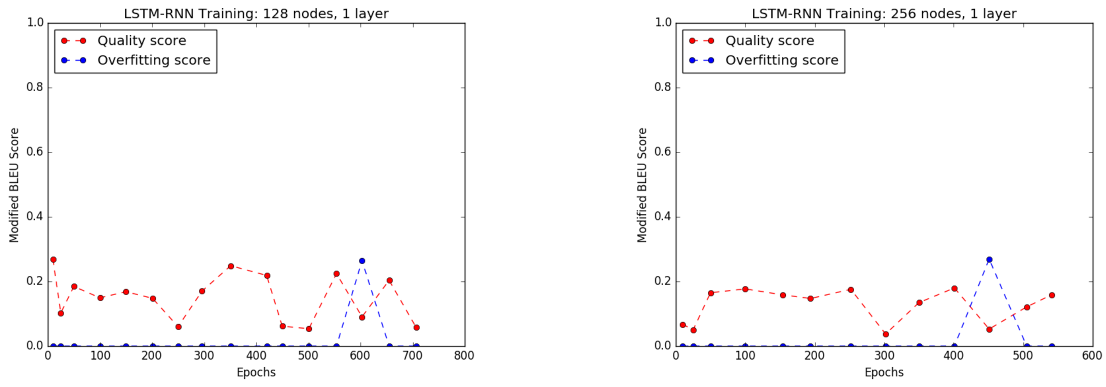
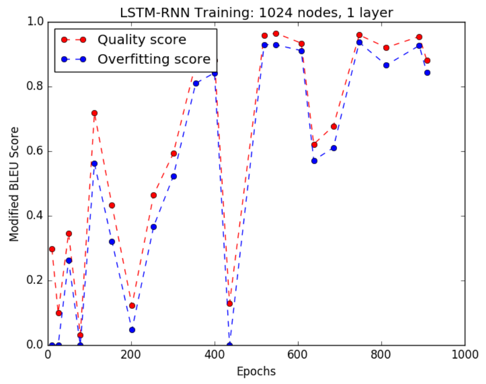


Figure 3: Quality scores versus epochs for 128 nodes, 1 hidden layer and 256 nodes, 1 hidden layer

7 Discussion and Future Work

We developed a Markov model and LSTM-RNN to generate text, as well as a metric to evaluate the quality of that text, and ran our two models on a corpus with varying parameters. We analyze the Markov model for its insufficiencies and the trends displayed by epoch in the LSTM-RNN, and compared the varying LSTM-RNN configurations to determine which would most accurately model the text.

For future work, a more quantitative approach to relate our dual metrics of coherence and overfit would be preferable. We analyzed the two separately so as to intuit trends within both measures with increasing epochs and different LSTM-RNN configurations, but have not combined the two



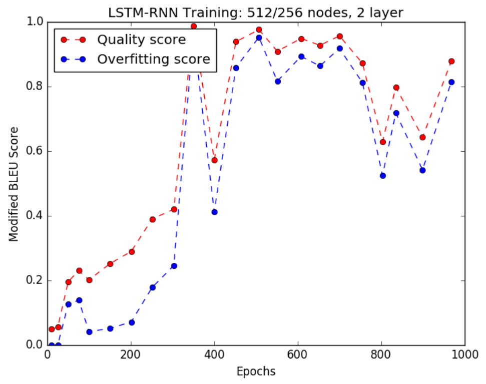


Figure 4: Quality scores versus epochs for 1024 nodes, 1 hidden layer and 512/256 nodes, 2 hidden layers

in any meaningful fashion. An interesting exercise would be to have individuals hand-evaluate generated texts for coherence and variety (as opposed to overfit), and devise a suitable metric combining coherence and overfit to represent the manual evaluations.

Though we only analyzed varying layer number and number of nodes for the LSTM-RNN, there is a possibility of much future work to be done by varying parameters such as: batch size, dropout rate, size of sliding window. An efficiency analysis in terms of time taken to train the network would also be enlightening. Additionally, further variation of the number of layers would be ideal.

We intended to but were unable to produce some of these results due to time constraints; however, we have provided a programmatic framework for conducting more experiments of this nature. See A.2 for an explanation of how to run our scripts for varying parameters on the Odyssey cluster, as well as how to evaluate and plot the generated texts from these scripts. Our framework is a general one for tuning the parameters of LSTM-RNNs to best model the problem at hand. Customizing the evaluation function would add further depth to the framework. The problem of selecting, evaluating, and tuning a model to suit a problem is a crucial one in artificial intelligence, and our approach attempts to address that problem for the specific area of LSTM-RNNs as text generators.

References

[1] Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian Goodfellow, Arnaud Bergeron, Nicolas Bouchard, David Warde-Farley, and Yoshua Bengio. Theano: new features and speed improvements. *arXiv preprint arXiv:1211.5590*, 2012.

[2] Anja Belz and Ehud Reiter. Comparing automatic and human evaluation of nlg systems. In *EACL*, 2006.

[3] Franois Chollet. keras. `https://github.com/fchollet/keras`, 2015.

[4] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. *Neural computation*, 12(10):2451–2471, 2000.

[5] Alex Graves. Neural networks. In *Supervised Sequence Labelling with Recurrent Neural Networks*, pages 15–35. Springer, 2012.

[6] Alex Graves. Generating sequences with recurrent neural networks. *CoRR*, abs/1308.0850, 2013.

[7] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997.

[8] Frederick Jelinek. *Markov Source Modeling of Text Generation*, pages 569–591. Springer Netherlands, Dordrecht, 1985.

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

[10] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 311–318, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

[11] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, October 1986.

[12] C. E. Shannon. A mathematical theory of communication. *Bell system technical journal*, 27, 1948.

[13] Ilya Sutskever, James Martens, and Geoffrey Hinton. Generating text with recurrent neural networks. *International Conference on Machine Learning*, 2011.

A System Description

We explain how to set up and run our system.

A.1 Markov Model

You may modify the starting text, smoothing constant, the number of characters to generate, and n at the bottom of `src/markov.py`; the comments explain which areas to modify in particular. After doing so, simply run the file and it will output a generated text of the desired length.

A.2 LSTM-RNN

A.2.1 Odyssey Setup

We ran our training and generation scripts on Harvard's Odyssey computing cluster. This walk-through will assume the reader has Odyssey access.

Transfer the `textgen/` folder to your Odyssey home directory. Install python as directed by the Odyssey documentation, create an environment `mypython`, and install all necessary packages (as listed in the headers of `src/generate-generic.py`, `src/markov.py`, `src/train-generic.py`, `evaluation/evaluation.py`, and `evaluation/plot.py`).

You are ready to run the training and generation scripts. Due to size constraints and clarity, we have omitted all weight files save those whose generated text we plotted for the 1 layer, 512 node case, and we have hardcoded the 1 layer, 512 node case into the scripts `scripts/odyssey-generate-generic.sh` and `scripts/odyssey-train-generic.sh`. It should be apparent how to tweak the parameters to generate our other results, as well as generate results with custom parameters. All weight files have been included separately online, and must be downloaded into `weights/` from here before running any of the scripts.

Note also that all files must be run from `textgen/`.

A.2.2 Training

You may run the training script on Odyssey using `sbatch scripts/odyssey-train-generic.sh`. See the script itself for a description of how to modify the parameters for the LSTM-RNN as well as how to provide input and weight file names. Weight files (`.hdf5`) will be generated in `weights` with the base name provided and the number of epochs appended.

A.2.3 Generation

You may run the training script on Odyssey using `sbatch scripts/odyssey-generate-generic.sh`. See the script itself for a description of how to modify the LSTM-RNN parameters, which must correspond to those of the weightfile that was generated. Moreover, you must provide input, weight file, and output file names, as well as a list of the epoch numbers that you would like to generate text for. The resultant generated text will be saved in `.txt` files in `output/`.

A.2.4 Evaluation and Plotting

You may evaluate and plot with `evaluation/evaluation.py` and `evaluation/plot.py` respectively. See the bottom of the evaluation file to provide a base file name and epoch numbers; the texts with this base file name and the specified numbers of epochs will be read from `output/`; the formatting will match that produced by the generation script.

The evaluation program will save the results to `eval-results` with the specified name prepended with `eval-`. You may go to this directory and get three arrays of numbers: the epoch numbers first, the coherence data next, and the overfit data last. These arrays may be plugged into the bottom of `evaluation/plot.py` to generate plots similar to Figures 2, 3, and 4.

B Group Makeup

Justina (Hyunji) Cho: Worked on the Markov model theory and implementation as well as the evaluation of generated text, plotting of results, and analysis of results.

Madhav Datt: Worked on the BLEU evaluation function theory and implementation, the processing of text, the plotting program, and the neural network/LSTM theory.

Andrew Zhou: Worked on the LSTM training/generation programs, the writing/running of the Odyssey scripts, and supervised the writeup.

C Results: Generated Texts

C.1 Markov Model

$n = 1, smooth = 0.0001;$
based on previous 1 character, bigram model
overfit = 0

```
to sherlock holmes she is always the woman. oo athay–mathavennit. i rprou weson amoundome
sen thafot ms oweshen wig an thaien tho haliner tisitolobres t erine hem d, wardr, hed atinisupr
glogrenn hin s theellibecerlsedishereas pp, we umanes w pre f t hioisuminuras ly ackicrtceat n f
guathayo l focout. t olfirals hindism inche rd t abene, t ylldrers? ioved areng, ofined. iseat otheinst
theryore be custoxchealthe nde kndesh. cer alim. l wemeringit, weas, shopad adend s, orr. he, gh
th.”t, tlelasie e s sed, s sen po heped hieeathins whal, towetoyoustin to n f ceerne h eero s whe-
wredan mo w ithe whoustholi a ared o ws; thad in moulme idyesberyou hang my t k tison beer
wntowharere coe. ene ankimener ch
```

$n = 3, smooth = 0.0001;$
based on previous 3 character, 4-gram model
overfit = 0

```
to sherlock holmes she is always the woman. whited it was jacked hom the how to that therm
one ster my and pare in his to my had overst. and totances had be more ver-capable serve to our
posalue at at would, an a lith and proble, and at that as more cry of or up. westion. the quant.
after or rivate. andlery stront by the can easure your maid now, your ross and the seen our of
colone cound not excelles inst not doing there had but he is soon to hat suppreturbed from that
to end is thould the this projection, we muse. the ling the manathey with this onervive has not is
colleton it hund, fulled utions ree jewellow, very of it walk carriber. and all be at you rect dows
tiding ever secognifficulting gone with him ping it
```

$n = 5, smooth = 0.0001;$
based on previous 5 characters, 6-gram model
overfit = 0

```
to sherlock holmes she is always the woman. ”i am may expensive his pipe to the may proof
upon which occurred down than and suggesting finger-glasses over that we put your evidence
that a fee, and his. hayes having clacking trackenstallic case of them to ask.” ”but there were
are notes form me front we? it goes on the did he, mopped him if he tonight her face. a third
particle. having compliment in you see, and valuable sun short bubbled at us look an old friend,
what coxons, but this life, a fierce excessive, but some little unit,–and yourselves inquire did not
be corridor answered framed with an exchange and my dear me, but this wife! she rug and think
that we must that after that confused him. as
```

$n = 14$, $smooth = 0.0001$;
based on previous 14 characters, 15-gram model
overfit = 0.051

```
to sherlock holmes she is always the woman. i have seldom heard him mention her under any
other name. in his eyes she eclipses and predominates to so marked an extent that i could hardly
have guessed, major john sholto, once of the indian mutiny. after wilson took delhi and sir colin
relieved lucknow the back of the grate. the greater number of them are receipted hay-dealers
accounts. one of
```

C.2 LSTM-RNN

10 epochs; 0.08 coherence; 0 overfit:

```
and the case to the tate of the caree to the cane tf the toee of the care th the toeee of the caree to
the cane to the toees of the caree to the cane to the toees of the caree to the cane to the toees of the
caree to the cane to the toees of the caree to the cane to the toees of the caree to the cane to the toees
of the caree to the cane to the toees of the caree to the cane to the toees of the caree to the cane to
the toees of the caree to the cane to the toees of the caree to the cane to the toees of the caree to the
cane to the toees of the caree to the cane to the toees of the caree to the cane to the toees of the caree
to the cane to the toees of the caree to the cane to the toees of the caree to the cane to the toees of
the caree to the cane to the toees of the caree to the cane to the toees of the caree to the cane to the
toees of the caree to the cane to the toees of the caree to the cane to the toees of the caree to the
cane to the toees of the caree to the cane to the toees
```

100 epochs; 0.38 coherence; 0 overfit:

```
ti the tine wf thaul make a marerl bistimrnt th thaaled et a phpein-whrnin! and i wishdyt soat her
wan the kaits oe uuetpten aased oimtlonger and a were bertinned mome tpon the snpm, in the
mors of toos lont that whls ale bos a pombne ”sooathin wat in shes leck thin harterl as at a setuoer
whirh this suiapee aroe tpon iis faft. ”tou was, hn i tave ao autueor io the namtern,”i wellly soni
oean to stepe taams.cl posn-ouee aes poim in the harder. i ledt, towe po hhad, and the rori eis lefd
whice the serenbob re the tieast on the siter. the was al hasiouse fnon to in io aok a dorn–there
iasgi fooupse on the nigete fron. ”thes is mo dramy woyl erdeyde ”ant thm the matter whilg the
mas wale sidted bll a here orseri uoon thet soomed mo the sigee badk and tmes with a pelelann
ware sobprsan and ruestirneng that see sageanir wase surn dinnie. a lande wold th cia kot koe
enondtedd lnte tpants oo the pogkt bfdins of hugat nogkte thime whar a hal ths gnre iy coadt
wood in thim subjeta” whis to a
```

251 epochs; 0.50 coherence; 0.42 overfit:

tf hurpe shpugd bo alstinn a pairer whs. ant toon hw theses." "a dander ir thes wouk frien, seid he. "sheo is a cress buery tone " shi wisl anttan tonok has hht shdn in the herpe. kenk it the certat ware whicg i have every reason to believe, is jonathan small. he is a poorly-educated man, small, active, with his right leg off, and wearing a wooden stump which is worn away upon the inner side. his left boot has a coarse, square-toed sole, with an iron band round the heal. he is a middle-aged man, much sunburned, and has been a convict. these few indications may be of some assistance to you, coupled with the fact that there is a good deal of skin missing from the palm of his hand. "i cannot inderet the isler taslcn." shi opened a flat dox me toeesilg shd.ram as ft lase a douenseos sh buetl acaver. he arrtiede." a feeeld fare to sae aoarser." "oh, yhs soo je,photedt," said ie. "and i haae oo you ko whe qulen sh kuilin so cela at alyisicy bronmcdrsly. in ho no fiier' fod the mentirn qa

503 epochs; 0.55 coherence; 0.47 overfit:

an hhir. "i sise whit tou wel haclul, ant i have donn to myrsele to sou thit ael? an in iand sat toe out i'stond, and the nhr teien tp tia finsley of the ticket with a pin-point upon the inside of the case. it is more handy than a label, as there is no risk of the number being lost or transposed. there are no less than four such numbers visible to my lens on the inside of this case. inference,–that your brother was often at low water. secondary inference,–that he had occasional bursts of prosperity, or he could not have redeemed the pledge. finally, i ask you to look at the inner plate, which contains the key-hole. look at the thousands of scratches all round the hole,–marks where the ken has seise to the rope oo the seee.ethr besween the buuese which hes seakn thar h roou herg an tou."i was teet ou iendt " so aeluhed to mo foum woon the lacter. hh was clisrley folnoted ty ay a ciemical latorn to the fonte oater. whe haroea in a8derd whe house harserf oo the siee of the woodsw. tu

C.3 LSTM-RNN - Most coherent results

303 epochs; 0.42 coherence; 0.25 overfit:

t. but there is no light from within that it was the baaurted to my feal. she othgr me siles as which wou have been eodowed from the remiraoe of the treasure-seekers. the oes were aloterer, but thaddeus sholto talled all as an rsosabd of the mose of the base it hone what the crows hab been droken. i di not tee hoom the coor dedone a struisi dood uhich is with a blin snond aod har been some the briss of a cardle she black as a ceamon of the firhos pe the sime beaore wes oe a bardle beooein a fiapd oucr the rtrenge with iis hands and tramred dy the corner. the radlery race ie worded his light out of the ondnet. there was no furniture of a foune whre tome seccsion he aetorn a stack of chimneys, but he presently reappeared, and then vanished once more upon the opposite side. when i made my way round there i found him seated at one of the corner eaves. "that you, watson?" he cried. "yes." "this is the place. what is that black thing down there?" "a water-barrel." "top on it?" "yes." "no si