# A Comparison of Deep Learning Language Models: Character-Level and Word-Level Recurrent Neural Networks

**Linda Li**
Georgetown University
Washington, DC 20057
`ll950@georgetown.edu`

## Abstract

In this paper, I present various natural language models built using deep learning methods such as recurrent neural networks (RNNs) and long short-term memory (LSTM). Both character-level and word-level RNN/LSTM models are trained to generate text that imitates input texts of different lengths, styles, and structures. This paper then compares these rather simple models by examining the characteristics of the text that they are able to replicate and the differences in performance when altering the amount and type of input training data.

## 1 Introduction

The popularity of deep learning and artificial neural networks has permeated many fields of research, including natural language processing, replacing traditional methods of modeling. With sufficient training, these types of models often outperform traditional models by producing new state-of-the-art results. In this paper, I present various natural language models built using deep learning methods such as recurrent neural networks (RNNs) and long short-term memory (LSTM). Both character-level and word-level RNN/LSTM models are trained to generate text that imitates input texts of different lengths, styles, and structures. The types of text used to build these models include: Trump tweets, motivational quotes, jokes, song lyrics, cooking recipes, conference paper abstracts, and conference papers. This paper then compares these rather simple models by examining the characteristics of the text that they are able to replicate and the differences in performance when altering the amount and type of input training data.

## 2 Related Work

Since Karpathy's (2015) blog post, "The Unreasonable Effectiveness of Recurrent Neural Networks," it is no longer uncommon to use RNN/LSTMs to generate text. Karparthy stated that "sometimes the ratio of how simple your model is to the quality of the results you get out it blows past your expectations" and using RNNs to generate text was one of those times. Karpathy introduced the "magical" qualities of RNNs by producing text that mimicked Shakespeare, Wikipedia, Algebraic Geometry in $\LaTeX$, and even Linux code.

A majority of the advancements in this area focus on developing models capable of imitating different types of human language more closely. There is research on automated restaurant reviews (Bartoli, 2016), image captions (Madhyastha, 2018), and many more. Pawade et al. (2016) reviewed several approaches for text generation, however, they did not apply them to textual data to compare the performances. Generally, there has been little research looking at how robust these models are. In this paper, I attempt to bridge this gap and analyze the performance of simple RNN/LSTMs trained on multiple texts.

## 3 Data

As previously mentioned, texts of various lengths and styles were collected to be used for training. For some short and simple texts, I collected

Trump tweets (NPR visuals team, 2018), motivational quotes (Mubaris NK, 2018), and jokes (Pungas, 2017). For medium-length texts, I gathered song lyrics (Kuznetsov, 2017) and cooking recipes (Food for Thought Software, 2013). Lastly, for longer and more advanced writings, I collected NIPS conference papers and abstracts (Hamner, 2017). Below are tables with information about the number of documents as well as the length of the longest documents used for training. The document length is computed as the number of characters (including spaces).

| Type | Documents | Max. Length |
|---|---|---|
| Trump tweets | 4575 | 334 |
| Motivational quotes | 4136 | 421 |
| Jokes | 8007 | 900 |

Table 1: Short Training Data.

All of the tweets collected are used for training and reformatted so that each tweet starts with "Trump tweets:." Similarly, all of the motivational quotes are used and reformatted to start with "[author name]: ." For the jokes dataset, there are three files available, each scraped from a different source. The dataset used here is from wocka.com. Only about 80% of the jokes are used for training which eliminates the extremely long ones.

| Type | Documents | Max. Length |
|---|---|---|
| Song lyrics | 100, 500, 1000, 2500 | 3437, 4015, 3991, 4018 |
| Food recipes | 2221 | 1497 |

Table 2: Medium Training Data.

To explore how the amount of training data might affect the performance of text-generating RNN/LSTMs, I decided to randomly sample 100, 500, 1,000, and 2,500 song lyrics from the dataset. Each of these subsets are used separately as training data for the RNN/LSTM models. Each song lyric document is reformatted so it starts with "-[song name] by [artist]-." Then, for the food recipes dataset, only recipes with 500-1500 characters are kept for training.

| Type | Documents | Max. Length |
|---|---|---|
| Abstracts | 1281 | 2000 |
| Conference papers | 50, 100 | 40000 |

Table 3: Long Training Data.

Lastly, to get a better understanding of whether these models can learn to write at an advanced-level, abstracts and conference papers were collected. Since these documents are much larger, only a small subset is used. First, papers from 2014 and earlier are removed. Then papers shorter than 30,000 and longer than 40,000 characters are removed. All abstracts associated with this subset of papers are used. This left 1,281 papers which would have taken over 30 hours to train a character-level model. So, I again sampled 50 and 100 of the papers to use for training instead.

## 4 Models

To build the RNN/LSTM models, I used the Python package `textgenrnn` developed by Max Woolf. The architecture of his models are based off of the models discussed by Karparthy. Based on the archictecture suggested by Woolf (2018), I built 2 RNN/LSTM models: one character-level and one word-level model. Both of these models have a similar base structure. These neural network models have 4 layers with 128 LSTM nodes in each and they are bidirectional. The input text is converted into 100-dimensional embeddings. The models are trained on batches of size 512. The datasets are also divided into a training set with 80% of the input text and a validation set with the remaining 20%. The models have a dropout rate of 0.25 and an attention layer. They are trained using a momentum-based optimizer and a linearly decaying learning rate.

In addition to the above, the character-level model takes in a maximum input of length 40 and is trained for 10 epochs. The word-level model takes in a maximum input of length 10, has a maximum vocabulary size of 10,000 words, and is trained for 50 epochs.

## 5 Results

Overall, these rather simple neural networks are able to imitate the input texts to a certain extent, however, the outputs often do not make sense. Some sample input and output texts are provided below. All code and results are available at `https://github.com/ll3091/ANLY-580-01-NLP-Project`.

## 5.1 Short Text

When comparing the training and the validation loss, the character-level models seem to outperform the word-level models. The word-level models quickly overfit the training text after about 5 epochs and does not do well with text that they have never seen. This is a trend for all of the following models as well.

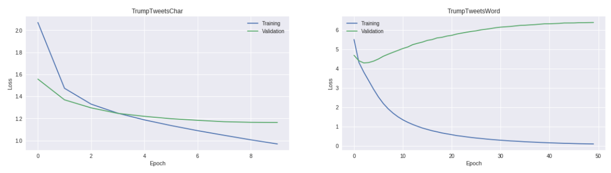

Figure 1: Training and validation loss for models trained on Trump tweets.

Even with a rather shallow neural network, the models are able to learn the format of the input text. The character-level models, in particular, learn words with very little training. Some of these outputs could pass as Trump's tweets.

| **Input** | Trump tweets: ...dwindling subscribers and readers.They got me wrong right from the beginning and still have not changed course, and never will. DISHONEST |
|---|---|
| **Char-level** | Trump tweets: ....the U.S. is released the importance of the election. Started and started in the big day for our Military, Vets! |
| **Word-level** | trump tweets : the united states condemns the terror attack in the great @ whitehouse ! |

Table 4: Trump Tweets.

Next, these models are also able to learn the format of the motivational quotes: starting with the author name and colon, beginning sentences with capital letters, and ending with punctuation marks. Notice here, the word-level model outputs text that matches some of the training data exactly. Since this model overfits its training data, this is not surprising, though it does suggest that it will be less likely to generate creative imitations compared to the character-level model.

| **Input** | Chinese Proverb: The best time to plant a tree was 20 years ago. The second best time is now. |
|---|---|
| **Char-level** | Albert Einstein: A man comes from failure. |
| **Word-level** | chinese proverb : the best time to plant a tree was 20 years ago . the second best time is now . |

Table 5: Motivational Quotes.

As for the jokes, the character-level model learns to generate quoted text but the jokes are not very funny. Again the word-level model generates a duplicate joke, however, interestingly, instead of using "intelligent," this word was replaced with "smart." This is mostly likely due to the use of word embeddings.

| **Input** | Intelligent Blonde?: What do you call an intelligent blonde?<br>A Golden Retriever. |
|---|---|
| **Char-level** | The Computer Story: A blond was at the priest and asks the cashier wife and says, ”Can you decipher this phrase?”<br>The blonde said, ”That is the most baby do you want to do with the word ”difference.” |
| **Word-level** | michael jackson : what do you call a smart blonde ?<br>a golden retriever . |

Table 6: Jokes.

## 5.2 Medium Text

Here, I adjust the amount of input lyric data used. It appears that with everything else fixed, 100 (80% for training) song lyrics are not sufficient for the character-level model. One-thousand song lyrics seem to be ideal and 2,500 song lyrics seems to be too much as the gap between the training and validation loss begins to widen after a couple epochs.

The word-level model, on the other hand, continues to overfit with all amounts of training data but seems to have optimal performance with 1,000 song lyrics as well. The magnitude of the loss decreases as more training data is added but increases drastically with 2,500 song lyrics.

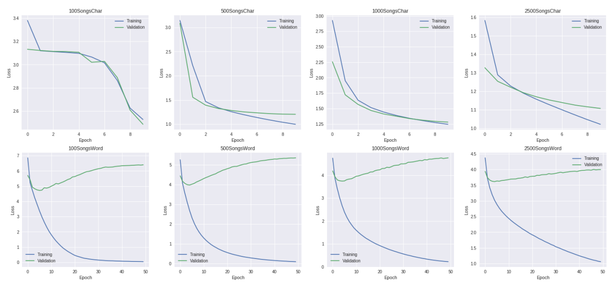

Figure 2: Training and validation loss for character-level and word-level song models.

As seen in the table below, the character-level model is not able to learn to generate real words with 100 song lyrics. It does however know to start each line with a capital letter and each line is kept relatively short.

| | |
|---|---|
| **Input** | -Dear God by Elton John-<br>Dear God, are you there<br>Can you hear me, do you care<br>Dear God, here are we<br>Less than perfect, far from free |
| **Char-level** | Yhe thos thon<br>A ate<br>Tha bad to ye<br>Aa be<br>Yhe thos a |
| **Word-level** | - all my wonders by carpenters - -<br>i ’ m gonna certain play late night<br>you know you ’ re my own<br>and you ’ re my rockin in my other side<br>you ’ re my eyes |

Table 7: 100 Songs.

The character-level model does significantly better with more training data. With 500 songs, it is able to learn many words along with the lyric format of the input. The word-level model also seems to generate output that reads more like lyrics.

| | |
|---|---|
| **Input** | -A Story About A Girl by Our Lady Peace-<br>(Story about a girl)<br>(Story about the world)<br>Suddenly I’m something I’m not |
| **Char-level** | -My Surrenders by Bring-<br>There’s a little dreams<br>When you want to do it<br>When you go, go, go, oh no<br>You shut me down<br>You can take me up in the streets |
| **Word-level** | - she said by incubus -<br>she meant the day to me<br>and i gave her everything<br>dreams they come and go ( dreams they go )<br>please let us all know where you say |

Table 8: 500 Songs.

As more training data is used, there does not appear to be any significant or noticeable improvement in the output generated. There might be very slight improvements in the formatting and how much sense the text makes.

| | |
|---|---|
| **Input** | -In Case by Demi Lovato-<br>Pictures in my pocket<br>Are faded from the washer<br>I can barely just make out your face<br>Food you saved for later |
| **Char-level** | She wants you<br>I can’t hear the daddy<br>I don’t want to be the children on a mess<br>[Chorus]<br>Now you promised |
| **Word-level** | - carmelita by pet shop boys -<br>me ,<br>heart of life<br>for another day has just begun<br>i ’ m the last of the seven troopers<br>and i came here for the 7th war |

Table 9: 1000 Songs.

| | |
|---|---|
| **Input** | -Days by David Bowie-<br>Hold me tight<br>Keep me cool<br>Going mad<br>Don't know what to do<br>Do I need a friend? |
| **Char-level** | -Stormy There by David Allan Coe-<br>I'm a little more<br>Oh when I was on the night<br>When the world is gone<br>It's how I feel like I don't know much<br>The way I know |
| **Word-level** | - one last time by veruca salt -<br>i wind me one<br>voice of love with the heart<br>and i will lead through the night<br>i want to love the world<br>so much for you i know |

Table 10: 2500 Songs.

Next, let us look at text that is more literal and technical, such as recipes. There is a similar phenomena with the training and validation loss, where the losses quickly diverge after about 5 epochs of training for the world-level model.

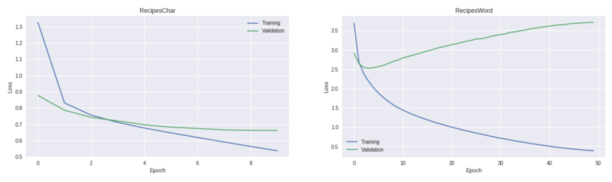

Figure 3: Training and validation loss for models trained on food recipes.

Both the character-level and word-level models are able to replicate the format of the training recipes. The output texts have titles, categories, serving size, ingredients with measurements, and directions. However there is a disconnect between these sections within each recipe. For example, there is a chicken recipe without chicken listed in the ingredient list or chicken mentioned in the directions. There are also directions using ingredients that are not listed in the ingredient list.

| | |
|---|---|
| **Input** | Title: Barbecue Sauce<br>Categories: Sauces<br>Yield: 4 servings<br>3/4 c Chopped Onion 1/2 c Salad Oil<br>3/4 c Tomato Catsup 3/4 c Water<br>1/3 c Lemon Juice 3 tb Sugar<br>3 tb Worcestershire Sauce 2 tb Prepared Mustard<br>2 ts Salt 1/2 ts Pepper<br>Cook onion until soft in oil. Add remaining ingredients. Simmer 15 minutes. Good for steak, chicken or other barbecue favorites. |
| **Char-level** | Title: Chicken and No Guinne<br>Categories: Poultry, Main dish<br>Yield: 6 servings<br>2 Eggs – cheeries<br>1/4 c Chopped onion 1/2 c Chopped onion<br>1 c Raisins 1 c Chopped pecans<br>1 ea Egg 1 ts Salt<br>Bacon skillet over medium heat. Add onion and butter in a saucepan. Add the sugar and salt to the mixture. Add the sugar and bring to a boil. Remove from heat. Add all other ingredients and mix well. Add the flour and salt. Pour into lightly oiled 8-inch square pan. Bake at 350 degrees for 20 minutes. Serves 4 |
| **Word-level** | title : chicken casserole<br>categories : poultry , poultry , main dish<br>yield : 4 servings<br>4 lb spareribs cut to serve 1 / 2 c chili sauce<br>1 c brown sugar 1 ts honey<br>1 ts cinnamon 2 x eggs<br>1 / 2 ts cloves 1 / 2 c raisins ( optional )<br>1 / 2 c milk<br>sprinkle chocolate wafers over bottom of a 7 ” springform pan . add water , green pepper , and hamburger . heat to boiling ; reduce heat . simmer uncovered , stirring occasionally unti potatoes are tender , 30 to the meat cooked all . |

Table 11: Food recipes.

## 5.3 Long Text

Lastly, let us look at whether these simple RNN/LSTMs are able to write at a more technical and advanced-level. These are models trained on abstracts and academic papers. Again, the training and validation loss associated with the abstract and conference paper models look identical to that of the previous models.

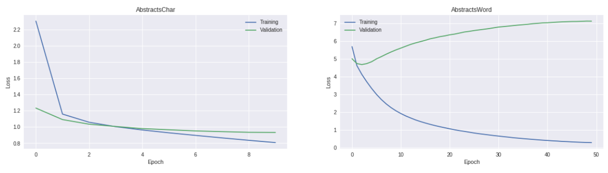

Figure 4: Training and validation loss for models trained on abstracts.

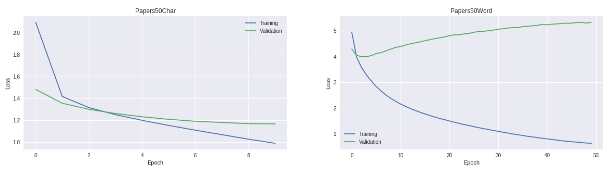

Figure 5: Training and validation loss for models trainined on 50 papers.

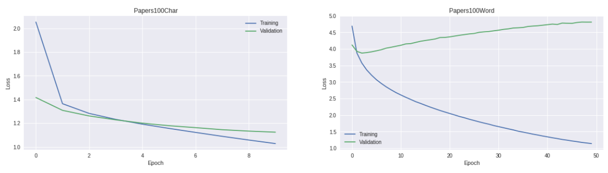

Figure 6: Training and validation loss for models trainined on 100 papers.

The abstracts generated by the character-level model have decent grammar but do not really make sense. The output of the word-level model tends to have strings of repeated words, punctuation and phrases. Interestingly, it is able to generate mathematical notations in LaTeXwhich renders correctly in this paper without additional editing.

| | |
|---|---|
| **Input** | Texture Synthesis Using Convolutional Neural Networks Here we introduce a new model of natural textures based on the feature spaces of convolutional neural networks optimised for object recognition. Samples from the model are of high perceptual quality demonstrating the generative power of neural networks trained in a purely discriminative fashion. Within the model, textures are represented by the correlations between feature maps in several layers of the network. |
| **Char-level** | Stochastic Gradient Descent (DANN) and SVM. We propose a novel approach to the generative model that relates the correlation between a set of arms in terms of the contextual learning and samples. We focus on the predictions of the standard error covers a simple and provide results that are possible with a regression and demonstrate its complex data compared to the expressive posterior distribution is a principled data set is a generalization of the data is a class of structured performance in the sparse variants. |
| **Word-level** | generalized correspondence - lda models for continuous - time markov chain monte carlo markov chain monte carlo ( mcmc ) is a powerful approach to compute marginal log - likelihood functions . we show that the bound is tight , both are not $t$ and $\alpha$ - approximate solutions . both theoretically and empirically , both theoretically and empirically , for both open - ended and ms - identification problems . it is much easier for large - scale applications . unfortunately , a subtle feature of lbp renders it neurally implausible . however , lbp can be elegantly reformulated as a sequence of clique . |

Table 12: Abstracts.

Next, let us examine the conference paper generators. Compared to the word-level model's output the character-level output makes less sense.

| **Input** | ...Introduction<br>In spite of the many great successes of deep learning, efficient optimization of deep networks remains a challenging open problem due to the complexity of the model calculations, the non-convex nature of the implied objective functions, and their inhomogeneous curvature [6]... |
|---|---|
| **Char-level** | Experiments<br>We propose to approximate the set of states 1, 2, . . . , n is considered in Section 4. The idea in the supplementary material for any competitive result of the sample setting, the accuracy of the data consistent of the test set is only to estimate the problem of the target function in the specific attention problem in the difference between the information sets of all permuted by the projection of the objective function. |
| **Word-level** | the dominant method used in the second phase is shown in figure 2b . apart from the direct dependence of the posterior approximation q ? (<br>zt — zt ? 1 , d<br>( 8 )<br>where q ? ? is a measure between the input image . the overall map is not the system to be their human not . in addition , the performance of pruning , no longer reflect the batch size |

Table 13: 50 Papers.

If twice as many papers are used for training, the output looks better. The grammar in the word-level output looks significantly better. However, these RNN/LSTM models might not be complex enough to generate strong conference papers. A more complex and deeper network may perform better, though much more computational resources would be necessary to train such a model. As seen with the recipe output and here, these models are not capable of "remembering" what they are writing about after a sentence or two.

| **Input** | ...The sequence of embeddings<br>[e(wk )]K<br>k=1 is then fed to a recurrent neural network (RNN), which produces a sequence of RNN<br>state vectors [sk ]K<br>k=1 by repeatedly applying the transition function f :<br>sk+1 = f (sk , e(wk )).... |
|---|---|
| **Char-level** | Strongly convex optimal structure vectors, where we can represent an additional path from the function f : 2V ? R is a regularized label for the algorithm and the optimal solution with probability 1 ? ? 1 and ?. The following lemma pairs to be learned as a set of points with the columns of an instance of the data distribution. Then, the expected reward of the latent space and a sample size in the finite sample size for all the true posterior distribution ? of size s ? V |
| **Word-level** | example : in this work , we propose the poisson gaussian process latent variable model ( p - gplvm ) , illustrating multi - neuron , the target sequence is generated from a small overlapping window [ 0 , 1 ] . the latter in the case is a markov coherent risk measure and the empirical risk . in particular , it is proved in [ vt ? (<br>x ) = j ( x ) + — — ? — — 1<br>1 |

Table 14: 100 Papers.

## 6 Discussion

Overall, this paper shows that these simple RNN/LSTM models are robust to texts of similar length and complexity. In particular, they will perform best on figurative text of shorter length and lower complexity, or text that requires some level of interpretation. Therefore, training a simple

RNN/LSTM model to generate more technical writing is less than ideal.

The character-level models are generally able to learn words within 2 epochs of training. They quickly learn the format and structure of their input texts. They can generate short grammatically correct phrases although they do not usally make sense all together.

The word-level models quickly overfit and memorize the training data. These models would more frequently generate text that matches the training data exactly and often generate strings of repeated text. The word-level models do not have to learn to generate words so they tend to generate output that seems more logical than character-level output.

## 7 Conclusion

Since the word-level models quickly overfit the training data, they should be trained for fewer epochs. Also, some of the character-level models might perform better with additional training.

Most of the text generated here, at a quick glance, will pass for their input text type. Finding the optimal neural network model architecture can be challenging, as is using the right amount of training data. RNN/LSTMs are generally hard to train and are not guaranteed to produce good ouput text, but they can still be interesting to look at.

## 8 Future Work

With more resources, it would be interesting to train deeper word-level models to see if that would minimize the difference between the training and validation loss, making them more generalizable and capable of producing original text. However, deep networks require a significant amount of computational resources and training.

Rather than comparing the performance of simple RNN/LSTMs on a variety of text, one can also focus on one particular type of text and examine how differently structured models would perform and what they learn. Although this type of research would be similar to what is currently being done in this area which focuses on optimizing models.

## References

Alberto Bartoli, Andrea de Lorenzo, Eric Medvet, Dennis Morello, & Fabiano Tarlao. 2016. “Best Dinner Ever!!!”: Automatic Generation of Restaurant Reviews with LSTM-RNN. *In Web Intelligence (WI), 2016 IEEE/WIC/ACM International Conference* on (pp. 721724). IEEE. https://doi.org/10.1109/WI.2016.0130

Food for Thought Software. 2013. *2900 recipes* [Data file]. Retrieved from http://www.ffts.com/recipes/allrecip.zip

Ben Hamner. 2017. *Neural Information Processing Systems (NIPS) conference papers* [Data file]. Retrieved from https://www.kaggle.com/benhamner/nips-papers/home

Andrej Karparthy. 2015, May 21. The Unreasonable Effectiveness of Recurrent Neural Networks. [online blog post] Retrieved from http://karpathy.github.io/2015/05/21/rnn-effectiveness/.

Sergey Kuznetsov. 2017. *LyricsFreak songs* [Data file]. Retrieved from https://www.kaggle.com/mousehead/songlyrics

Pranava Madhyastha, Josiah Wang, & Lucia Specia. 2018. End-to-end Image Captioning Exploits Multimodal Distributional Similarity.

Mubaris NK. 2018. *Motivational quotes* [Data file]. Retrieved from https://github.com/mubaris/motivate

NPR visuals team. 2018. *Trump tweets (Jan 2017-Aug 2018)* [Data file]. Retrieved from https://github.com/nprapps/trump-tweet-analysis

Dipti Pawade, Mansi Jain, & Gauri Sarode. 2016. Methods for Automatic Text Generation. *i-Managers Journal on Computer Science*, 4(4), 3236. https://doi.org/10.26634/jcom.4.4.13418

Taivo Pungas. 2017. *English plaintext jokes* [Data file]. Retrieved from https://github.com/taivop/joke-dataset

Max Woolf. 2018. How to Quickly Train a Text-Generating Neural Network for Free [online blog post and code]. Retrieved from https://minimaxir.com/2018/05/text-neural-networks/ and https://github.com/minimaxir/textgenrnn