The Divine Comedy—
A Comparison of Word-based and Character-based Text Generation Approaches

Daniel Halteh

Source Code— https://github.com/MSIA/dsh434_msia_text_analytics_2020/tree/final_project

Abstract

This paper details a comparative analysis of Natural Language Generation with both character-based and word-based approaches. Using a relatively small source text—Dante's *The Divine Comedy*—the Long short-term memory (LSTM) Recurrent Neural Network (RNN) performs best using the Word-based approach to tokenization and model training.

1 Introduction

The Divine Comedy, a long Italian narrative poem completed by Dante Alighieri in 1320 (just one year before his death), is widely recognized as one of the most significant pieces of literature in Italian history. Written in the Tuscan dialect, the Divine Comedy and its popularity helped establish the dialect as the official language of the Kingdom of Italy upon its formation. *The Divine Comedy* details Dante's journey through Hell, Purgatory, and Heaven, each of which forms a separate portion of the overall text. The goal of this paper is to learn to produce brand new Dante inspired text using *The Divine Comedy* as a source language.

2 Related Work

The first related text studied was a paper detailing a number of Natural Language Generation (NLG) tasks, approaches, and evaluation methods (Gatt et al., 2018). The authors propose using a Long short-term memory (LSTM) network for most NLG tasks, as well as suggest numerous qualitative methods for evaluating the performances of these models.

Another study was consulted to analyze the expected performance of both word and character-based NLG approaches (Bojanowski et al., 2015). This paper details the difficulties and limitations associated with using a character-based approach to text generation and provides alternative, hybrid solutions. Ultimately, the authors conclude that the gap between word-level and character-level models is fairly sizeable, stressing that the former remains the optimal modeling approach in most instances.

For properly evaluating the results of NLG models, further research supports the use of manual human evaluation as an alternative to common metrics such as BLEU (Xie, 2018). Although these metrics are successful in many Machine Translation tasks, they often do not properly capture linguistic fluency and coherence.

Lastly, two machine learning blog posts sourced from Jason Brownlee's book *Deep Learning for Natural Language Processing* were also consulted (Brownlee, 2020). This post takes a more applied approach to developing LSTMs by providing practical advice for hyperparameter and model selection, tuning, and training.

3 Dataset

The dataset used in this paper comprises of the entirety of Henry Wadsworth Longfellow's English translation of Dante's *The Divine Comedy*. This translation was obtained using Project Gutenberg's online text repository, which offers a number of free media materials that are now part of the public domain.

The text itself is divided into three separate sections of the text, known as *cantiche*—Inferno, Purgatorio, and Paradiso—each of which corresponds to distinct part of Dante's journey. Each *cantica* is comprised of 33 *cantos*, which is the principal form of division within long-text poetry. In total, the pre-processed dataset contains 14,443 lines and 600,189 individual characters.

4 Method

This paper utilizes the same network structure (LSTM), but serves to compare the tokenization and context approaches to how the language itself is analyzed and predicted. Namely, the two approaches compared here are character and word-based NLG.

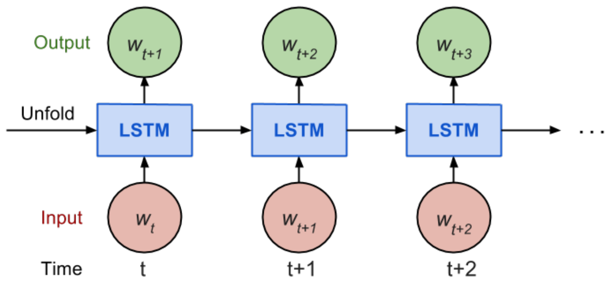

Figure 1—LSTM Architecture (Prasad, 2018)

4.1 Long Short-term Memory (LSTM)

The model of choice used in this paper is the Long short-term memory (LSTM) Recurrent Neural Network (RNN). The *Survey of the State of the Art in Natural Language Generation: Core tasks, applications and evaluation*, mentioned in the Related Work section of the paper, recommends using LSTM for a number a wide variety of text generation tasks. LSTMs aim to improve upon RNNs by avoiding that latter's issues with vanishing and exploding gradients by selectively choosing what to keep in long-term memory using "gates." These gates are defined in three varieties—

Gate	Purpose
Write	Store information
Read	Retrieve information
Keep	Designate which information to keep/forget

Table 1—Summary of LSTM Gates

4.2 Character-based Approach

The first approach implemented was character-based. This approach required treating the entire text as a sequence of tokens, each of which is represented by an individual textual character. The tokenized, post-processed text resulted in a vocabulary of 43 unique tokens, each of which corresponds to either an individual alphabetic character or some context-inclusive punctuation. Ultimately, the goal with the character-based approach is to model the relationship between characters within a given text, and use said model to generate new text based on a given character seed.

One advantage associated with using a character-based approach to tokenization and analysis is the increased potential for capturing meaning and structure within language, particularly with languages that have rich morphology. If this paper were utilizing the original Tuscan text, perhaps this advantage would lead to improved performance—but since this paper is instead using a translation in English (a language without a rich morphology), this may not be the case.

Another advantage to consider with the character-based approach is the potential for a well-performing model to generate combinations of characters that are not found within the original text. From a novelty perspective, it would be interesting observe how an LSTM interprets the combination of characters found within the English language, and more importantly how the model learns to create new meaning using said combinations as a learning source.

One of the glaring drawbacks of using a character-based approach is that said approach often requires larger corpuses or source texts to learn properly. Given that the focus of this paper's analysis is to create language using the *Divine Comedy* as an inspiration, the performance of the character-based approach will likely be hindered and/or limited by the relatively small size of the source text at hand.

Another drawback to consider with character-based approaches is the vast amount of resources need to properly model a character-based LSTM. Character-based LSTM models are highly computationally expensive in that they require much bigger hidden layers to successfully model long-term dependencies between characters, which means that overall training process tends be slow and arduous.

4.3 Word-based Approach

The second approach implemented was word-based. This approach required treating the entire text as a sequence of tokens, each of which is represented by an individual word within the text. The tokenized, post-processed text resulted in a vocabulary of 12,565 unique tokens or words. The goal with the word-based approach is to model the relationship between tokens in the text in an effort to learn to generate newfound sequences of tokens based on a given input seed of text.

The main advantage of the word-based approach when compared to the character-based approach is the significantly lower degree of computation required to properly train an LSTM that performs well. This is because word-based models are better able to capture long-term dependencies than character-based models, a difference which will be stark and likely exacerbated by the relatively small size of the source text that is *The Divine Comedy*.

4.4 Embeddings and Padding

To handle embeddings specific to *The Divine Comedy* as well as to avoid the high dimensionality of standard one-hot encodings, the Keras Embedding class is implemented, which essentially learns the vector representations of the input text during the training process itself through what's known as an "Embedding Layer." This embedding layer takes in one-hot vector representations of a given sequence, and then attempts to map them down to a sub-dimensional set of representations. The weights themselves learn iteratively through backpropagation, which will ideally begin to capture the relationships/similarities between the tokens. For the Keras Embedding layer to function properly, input sequences are limited to a given length and are padded with zeros if necessary.

4.5 Hyperparameter Choices

LSTMs come with a number of different hyperparameter choices than are essential for producing an optimal model.

The first parameter examined was the number of memory units to use within the LSTM layer of the network. Since character-based LSTMs require much larger networks to properly capture long-term dependencies between the characters themselves, the network was trained using both 256 and 512 memory units for purpose of comparison.

The second parameter considered is dropout, which randomly eliminates nodes in each layer for each training example or pass. This serves as a regularization technique and prevents a given model from relying on a select set of features since there are many passes in which they can essentially disappear. From a model training perspective, leveraging dropout will improve performance by preventing the model from overfitting. For network training and evaluation, dropout parameters of 0.10 and 0.20 were chosen.

Another parameter that was examined was the optimization parameter. Using an optimization algorithm helps speed up the learning process and prevents against model underfitting. For this paper, the Adam optimizer is used due its documented success.

When training the models themselves, another parameter to consider is the number of epochs required. Training models for a larger number of epochs helps reduce underfitting, too many epochs can directly cause a model to overfit the data. Since evaluating underfitting and overfitting on text generation tasks is quite difficult, the approach taken in this paper is to monitor the categorical cross-entropy loss of the models during training while also using a post-training evaluation process that relies on a qualitative check to determine the performance of the model itself on actual text generation tasks.

5 Results

Analyzing the performance of Natural Language Generation models is fairly difficult and subjective. Related research—(Gatt et al., 2018) and (Stanford, 2020)—suggest human judgements as a viable method for evaluation. Although fairly subjective, such a qualitative method for evaluating the performance of the models does make sense given the qualitative nature of the output itself and the text generation task at hand.

5.1 Character-based Results

To gauge the success of the character-based model, three separate models were trained varying both the number of memory units in the LSTM layer as well as the dropout probability parameter. For each model, the final categorical cross-entropy loss after 10 epochs is shown in the table below.

Memory Units	Dropout Parameter	Final Loss*
256	0.1	2.52
512	0.1	2.08
256	0.2	2.47
512	0.2	2.26

Table 2—Results from character-based LSTM
**Final loss stems from training LSTM for 10 epochs*

Although the loss values look consistently small, the a qualitative review of the performance of the four models demonstrate that the character-based approach does not perform well. Using the LSTM with 512 memory units and a dropout probability parameter of 0.1, for example, the following text was generated using the seed “my frozen body:

Input Seed	Output
my frozen body	was a saieo saieo saieo

Table 3—Generated text from character-based LSTM

As shown in the output above, the generated text is far from anything intelligible. Even with providing the model with a sequential character input that stems from the source text itself, it is evident that the character-based LSTM has not properly captured the character dependencies, and thus is not a viable solution for natural language generation sourced from a relatively small text.

5.2 Word-based Results

To gauge the success of the word-based model, four separate models were trained varying both the number of memory units in the LSTM layer as well as the dropout probability parameter. For each model, the final categorical cross-entropy loss after 50 epochs is shown in the table below.

Memory Units	Dropout Parameter	Final Loss*
128	0.1	2.78
256	0.1	1.64
128	0.2	2.92
256	0.2	1.80

Table 4—Results from character-based LSTM
**Final loss stems from running training algorithm for 10 epochs*

After comparing the outputs of each model above—using the same text seed provided to the character-based models for consistency—the LSTM with 128 memory units and a dropout probability parameter of 0.1 generated the most logical output. A few other sample inputs and outputs are shown below for further analysis.

Input Seed	Output
my frozen body	was in the glory of the hearts
free me from	out of the old world
how much	as I might hear for fear I do not say
how deep was	the affectionate appeal by its own place

Table 5—Generated text from character-based LSTM

As shown in the output above, the generated text from the word-based LSTM much more closely resembles the language from *The Divine Comedy*. While the text itself is not perfect—and admittedly at times semantically meaningless—the structure of the output is fairly close to that of Dante's work. This allows us to conclude that the word-based LSTM does seem to be a viable option for capturing the dependencies within the text and can successfully produce Dante-inspired text.

6 Discussion

To further evaluate the efficacy of word-based LSTM's at generating Dante-like text, four sample outputs from both *The Divine Comedy* as well as the artificial generator were presented to five individuals—each person was asked to rate the output on a scale from 1-10 in terms of said person's level of confidence (with 10 representing the highest level of confidence) of the output being an actual excerpt from Dante's *The Divine Comedy*. Although each individual was presented with two real and two artificial texts, this information was not disclosed to avoid any potential comparative bias. The aggregated results are shown below—

Mean Confidence on True Text*	Mean Confidence on Fake Text*
9.15	3.20

Table 6—Results from Qualitative Evaluation
**Confidence ranges on an ordinal scale from 1-10*

Although this paper is able to validate that word-based approaches outperform character-based approaches for text generation—particularly in the case of a small source text—the results from the mini-experiment demonstrate that this model is far from accurately capturing Dante's writing.

Also, given that character-based approaches are particularly effective when given larger texts to learn from, leveraging a significantly larger source text—perhaps a collection of Italian literature—would likely increase said model's performance (this would likely apply to word-based models as well).

Other potential avenues for improvement include implementing a Bidirectional LSTM, which uses forward and backward connections within the hidden layers to improve upon standard LSTMs, as well as levering pretrained embeddings such as word2vec and GloVe to analyze their impact on performance relative to learned embedding layers.

References

Bojanowski, Piotr, et al. "Alternative Structures for Character-Level RNNs." *ICLR*, 19 Nov. 2015.

Brownlee, Jason. "How to Develop a Word-Level Neural Language Model and Use It to Generate Text." *Machine Learning Mastery*, 7 Oct. 2020, machinelearningmastery.com/how-to-develop-a-word-level-neural-language-model-in-keras/.

Brownlee, Jason. "Text Generation With LSTM Recurrent Neural Networks in Python with Keras." *Machine Learning Mastery*, 2 Sept. 2020, machinelearningmastery.com/text-generation-lstm-recurrent-neural-networks-python-keras/.

Gatt, Albert, and Emiel Krahmer. "Survey of the State of the Art in Natural Language Generation: Core Tasks, Applications and Evaluation." *Journal of Artificial Intelligence Research*, vol. 61, 30 Jan. 2018.

Prasad, Harshit. "GSoC 2018: RNN and LSTM Networks - Part II." *Medium*, Medium, 10 Aug. 2018, medium.com/@harshit.prasad/gsoc-2018-rnn-and-lstm-networks-part-ii-96661bf24442.

Xie, Ziang. "Neural Text Generation: A Practical Guide." *Stanford University*, 22 Mar. 2018.