Deep Subreddit Simulator

Shamya Karumbaiah & Rafael Lizarralde

Abstract

Reddit *is a social news aggregation website with user-created sections called subreddits that are themed around a particular topic. We trained various models on an example subreddit's posts to generate human-like new posts. We explored n-grams, deep neural networks, and Markovify, an out-of-the-box markov chain generator. We evaluated these models using human ratings on applicability and coherence of a subsample of posts generated. We observe that although Markovify scored the best, neural models and n-gram generated more original posts. Neural models also needed larger corpus and higher training time as compared to n-grams and Markovify. Neural models also benefited learning English language representation by pre-training on a large OANC dataset.*

1. Introduction

Reddit is an online forum that uses a crowdsourced system for ranking posts and links submitted by users. Users can "upvote" or "downvote" posts and comments, which are sorted by how recently they were made and how much "karma" (upvotes - downvotes) they have received. In addition to this ranking system, Reddit has many user-created sections (or "subreddits"), in which ranking is done separately, that are themed around a particular topic, such as "news", a particular video game, or even just general themes like "totallynotrobots", where users submit posts as if they were robots pretending to be humans. We are interested in Reddit due to the ability to quickly and easily acquire a corpus of text themed around a particular topic. Our task is to automatically generate human like posts with a model trained on the posts from a particular subreddit. Thus, a trained model is specialized only to a specific subreddit.

2. Related work

For the last year there has been a subreddit, SubredditSimulator[2], which uses Markovify[16] to generate posts and comments, which are then voted on by real users. However, Markovify works at the phrase level rather than word or character level, so produces output that is very similar to a small number of input sentences. While this tends to have fairly amusing results, it renders it incapable of producing more original text. For example, for the model trained on the "world news" subreddit, a recent post was "Elephants in Tanzania reserve could be making up to $90 billion for allegedly killing Cecil, a well-known lion".

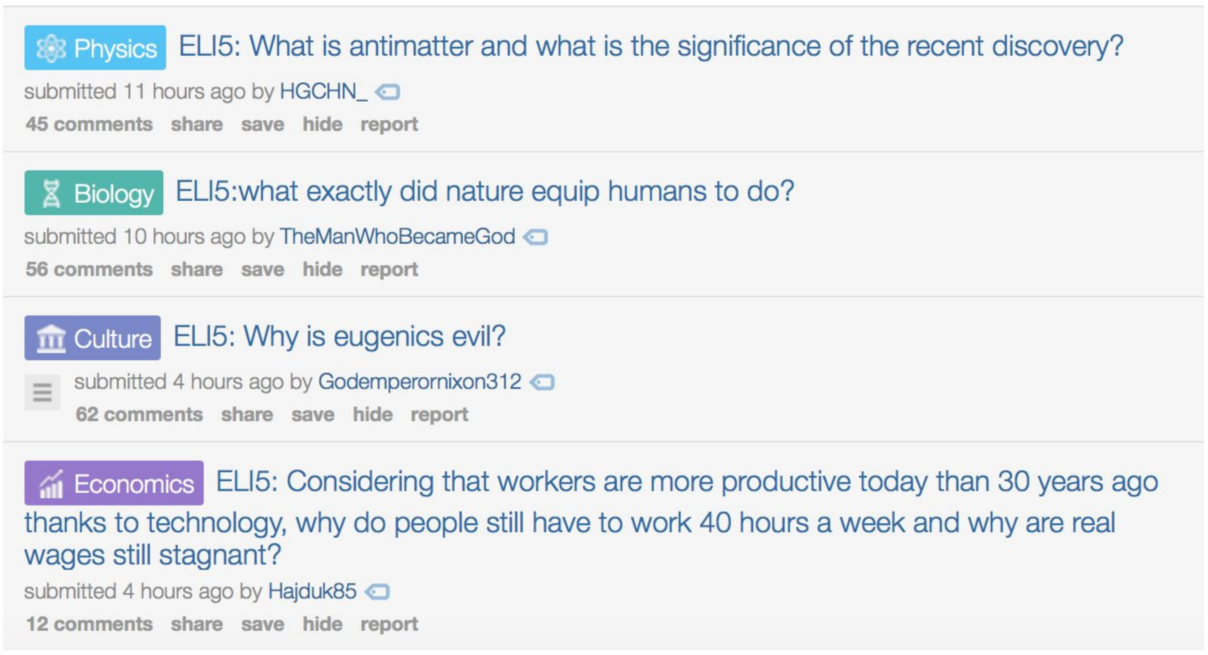
Character-level language models have some advantages beyond word-level language models, such as applications in text compression[14] and assisted typing. Their ability to model sub-word structure is also interesting, as it would be necessary to understand words made up during a conversation from morphemes (a property exploited by Lewis Carroll's *Jabberwocky* poem, for example); this type of wordplay is fairly common in some subreddits. Character-level language models can also be combined with word-level ones to make powerful ensemble models, which have won several text compression competitions[11].

There have been a variety of approaches to using recurrent neural networks for character-level language models. One of the key struggles for recurrent neural networks is the vanishing or exploding gradient problem; as error is backpropagated through time (backpropagated across the "unrolled" recurrent neural network), the gradients vanish (or in some cases explode). This makes it difficult for the network to learn long-term dependencies[1], which is even more of a problem for character-level models than word-level, as characters compose a longer sequence (and therefore more time steps of the recurrent neural network) for the same text. One approach to this has been developing alternatives to gradient descent, such as Hessian-free optimization[12]. Another approach is neural architectures that avoid vanishing gradients, such as Multiplicative Recurrent Neural Networks[17] and Long Short Term Memory (LSTM)[8]. This vanishing gradient problem also exists for deep non-recurrent neural nets, but has been mostly solved by careful initialization of the weights, such as by Xavier initialization[6].

3. Data

3.1 Choice of Subreddit

To demonstrate the possibility of an automated text generation, we chose to simulate the subreddit *Explain Like I'm Five*[13]. This subreddit allows its subscribers to post questions and topics they would like to get explained in simple language as if it were being explained to a five year old. Thus the post titles are usually short with a prefix *ELI5* (Explain Like I'm 5). Most of these titles are questions starting with question words such as "Why", "How", "What", "When", and ending with a question mark. There is no restriction on the domain and the questions usually ranges across economics, biology, physics, technology, culture, and more. See Figure 1 for some examples of posts in *Explain Like I'm Five*.


Physics ELI5: What is antimatter and what is the significance of the recent discovery?
submitted 11 hours ago by HGCHN_
45 comments share save hide report

Biology ELI5:what exactly did nature equip humans to do?
submitted 10 hours ago by TheManWhoBecameGod
56 comments share save hide report

Culture ELI5: Why is eugenics evil?
submitted 4 hours ago by Godemperornixon312
62 comments share save hide report

Economics ELI5: Considering that workers are more productive today than 30 years ago thanks to technology, why do people still have to work 40 hours a week and why are real wages still stagnant?
submitted 4 hours ago by Hajduk85
12 comments share save hide report

Figure 1: Example posts in the chosen subreddit *Explain Like I'm Five*

We selected this subreddit for the following primary reasons:

- High traffic which helps with relatively faster data collection
- Inherent syntactic and semantic structure that helps basic evaluation
- Shorter text length for easier evaluation

3.2 Data Collection

The goal with the data collection is to get a good distribution of the posts to represent the broad range of posts in the subreddit chosen, while avoiding posts that may not adhere to the subreddit's style guidelines. We used Python Reddit API Wrapper (PRAW)[4] to collect data from reddit, discarding posts that were less than a day old (so that all posts collected would have had time to be moderated). This python wrapper simplifies the access to Reddit APIs. A simple call to this API looks like:

```
r = praw.Reddit(user_agent='test_dss')
submissions = r.get_subreddit('explainlikeimfive').get_new()
```

This returns `submissions` as a list of objects with several attributes including post title and unique post identifier. A challenge with Reddit API is that it limits the number of results from every listing to 1000. As a workaround, we collected the top unique entries every day for a 30 days. See table 1 for a summary of the data collected.

Number of posts	3886
Number of tokens	74811
Number of types	8524
Number of sentences	4368
Avg length of a title	14 words

Table 1: Data statistics on posts from Explain Like I'm Five

4. Methods

4.1 N-gram

Our first model for text generation is the simple n-gram model. This model exploits the count statistics of the words in the training corpus to assign probabilities or weights to pick the most probable next word in the sequence. N refers to the size of the n-gram. A bigram model looks only at the last word, trigram looks at the last two words and so on. In general, to determine the next word x_i, the model looks at x_{i-1} x_{i-2} x_{i-3}... $x_{i-(n-1)}$ sequence of words. Thus, an n-gram model gives the probability:

$$P(x_i \mid x_{i-1}x_{i-2}x_{i-3}...x_{i-(n-1)})$$

Figure 2 shows an example of this model with dummy probability values used to demonstrate the model output. For example, since most of the posts in this subreddit starts with "ELI5", this model would give this token a very high probability compared to the other tokens.

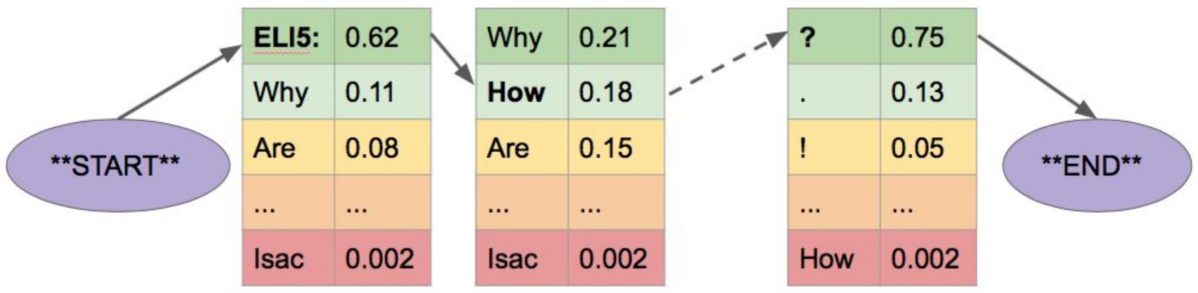


Figure 2: Visualization to demonstrate a dummy N gram model

4.1.1 Tokenization

The first step here is to accurately tokenize the training corpus to get the unique tokens. We used the Natural Language Toolkit (NLTK)[3] to achieve this using the python library *nltk.tokenize.punkt*. Based on the value of N, the model capacity increases. In the result section, we show that a higher value of N overfits the data and reduces the model performance by shortening the generated posts.

4.1.2 Count Statistics

We used python dictionaries to store the counts of the ngrams. A simple example of Ngram model counts with N = 4 looks as below -

*{**eli5:** : {why: { is: 12, are: 34, do: 8, ...}, how: {is: 14, are: 12, do: 4, .…...}*

…….}

…....}

This says that according to the training corpus, there 12 instances of the phrase "eli5: why is", 14 instances of the phrase "eli5: how are" and so on.

4.1.3 Weighted Selection

The count statistics give weights for the random draws of a new word given the n length sequence of the previous words.

$$P(x_i \mid x_{i-1}x_{i-2}x_{i-3}...x_{i-(n-1)}) = \frac{count(x_i,\ x_{i-1}x_{i-2}x_{i-3}...x_{i-(n-1)})}{count(x_{i-1}x_{i-2}x_{i-3}...x_{i-(n-1)})}$$

This approach has an inherent dependency on a moderately-sized training corpus that is representative of the true distribution of these tokens. This model also doesn't accommodate out-of-vocabulary words. Due to the tradeoff on the value of n, this model might suffer from long term dependencies in the generated text.

4.2 Markovify

Our second approach uses the out-of-the-box markov chain generator, *markovify.* It is similar to the N gram models explained above. One of the differences in this approach is in sentence splitting. *Markovify* remembers phrase level tokens and reproduces these phrases in the output. This is explained more in the results section. Given a large training corpus and the number of states, *markovify* generates a sequences of words, in our case truncated at the generation of a predefined end token.

4.3 Neural Models

For neural models we customized *torch-rnn*[10], an existing program for character-level text generation. *Torch-rnn* first preprocesses the input text to develop a character-to-integer mapping, and then uses one-hot vector representations of the characters as input for the neural net (Figure 3). We selected LSTMs (Long Short Term Memory) for our layer architecture as a way of getting around vanishing gradients, and also due to their dominant performance in many natural language tasks lately.

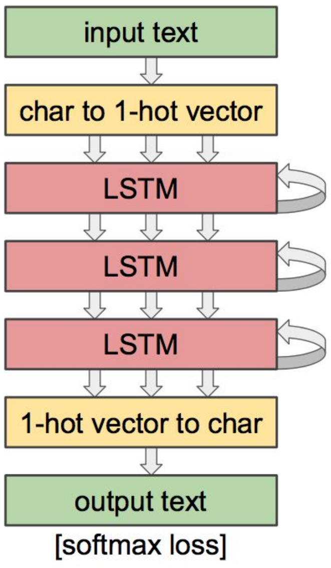


Figure 3: Neural model architecture

4.3.1 Training from Scratch

First we trained the model on the Reddit data that we had gathered for 400 epochs. However, the output indicated that the network was having trouble learning basic words and English syntax (although it learned the domain-specific syntax very well), so we also investigated pre-training it on a much larger corpus of English text.

4.3.2 Pre-Training and Fine-Tuning

We used the Open American National Corpus (OANC)[9], which contains almost 15 million words of English text in various forms (a novel, journals, a textbook, news articles, etc.). Because this was such a large data set, we were able to train for 6 epochs. We then trained the entire network for 300 epochs on our Reddit data set to fine-tune it to the Explain Like I'm Five domain.

5. Evaluation

Evaluation is one of the key parts of the project. Measuring the output quality in a text generation task is an ongoing research area. Since we don't have gold labels to evaluate the model against (such as you might have for captioning or summarization), doing an intrinsic evaluation is a challenge. We explored the text cohesion metrics, human evaluation, and extrinsic evaluation by posting output to Reddit. Our final evaluation is based on the second approach, human evaluation.

5.1 Measuring Text Cohesion

Text cohesion refers to the syntactic and semantic linking of the tokens in a sentence to give it a meaning. This is closely related to text coherence. As a first step to evaluate text cohesion of the generated posts, we tried the following widely cited tools:

1. TAACO[5] calculates 150 indices of local and global cohesion including basic indices like unique bigram, trigram counts, overlap indices and connectivities like sentence linking, conjunctions and order.
2. Coh-Metrix[7] measures the difficulty of a written text (mostly essays) to the target audience

Most of these tools focus on long texts like essays. By the nature of our domain, we tend to have very short posts. The posts are written in casual English, with loose adherence to traditional English grammar. These automated metrics focus on sentence structure and grammar and thus don't effectively evaluate the language used on Reddit.

5.2 Human Evaluation

Our second approach to model evaluation was to manually evaluate the model by annotating a sample of each model's output. For the four models, we randomly sampled 100 posts generated by them. The aim was to capture the text coherence and the relevance of the generated text to the subreddit chosen. We have used this approach to compare our models.

5.2.1 Annotation

The decision flow for the annotators is given in Figure 4, along with the description of each category and examples in Table 2.

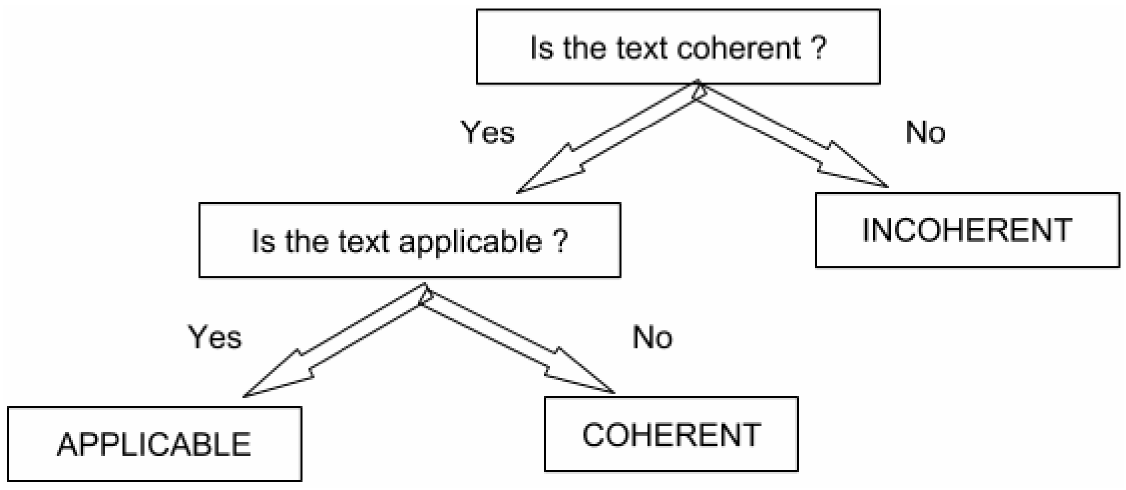


Figure 4: Decision tree for human annotations

Annotation	**Meaning**	**Example**
Incoherent	Rater couldn't understand meaning	*ELI5:are person just point of your own speakers differ in reaction?*
Coherent	Syntactically correct or made some sense	*ELI5: Why are some meats white while others are angry and mean?*
Applicable	Appropriate for /r/ExplainLikeImFive	*ELI5: What happens in the day with wire power lights?*

Table 2: Categories for human annotation

5.2.2 Inter Rater Reliability

Cohen's Kappa[15] is one of the strong measures of inter-rater reliability as it measures agreement as compared to chance. This is particularly important for us due to the skewed nature of our categories. For example, the neural model has only around 8% applicable rating. Since the expected agreement is already high (for example, a random rater that gives out "applicable" 8% of the time would achieve close agreement), the actual agreement must be higher to get a good Cohen's Kappa score.

Cohen's kappa is measured as the ratio of the proportion of times that the raters agree (A) to the maximum proportion of times that the raters could agree (both corrected for chance agreement (E)).

$$kappa = \frac{P(A) - P(E)}{1 - P(E)}$$

The value of kappa can vary from -1 to 1 where -1 denotes perfect disagreement, 1 denotes perfect agreement. A kappa of 0.4 to 0.8 is considered acceptable for two raters. The exact threshold depends on the amount of uncertainty involved in decision making in a particular dataset.

5.2.3 Mechanical Turk

We could extend the above approach of human evaluation to a larger scale using responses from Mechanical Turk. This was not affordable considering the scope of the project.

5.3 Extrinsic Evaluation

Extrinsic evaluation employs the output of the models in the task it is meant to cater. In our case, we could perform this by posting few examples of the auto generated posts to the *Explain Like I'm Five* subreddit and measure its performance with the upvotes it receives and with a qualitative analysis of the comments by the subscribers. It is hard to do an extrinsic evaluation at a scale representative of the performance of the model. Additionally, this may cause disruption in the community if there are too many posts submitted, particularly if many are of poor quality.

6. Results

6.1 Ngram models

Bigram models (n=2) performed the best. Higher values of n shortened the model output. We suspect the small training data size to be a reason for this as the count statistics trend toward zero with longer word sequences.

6.2 Markovify

Markov Chains with 3 states had qualitatively better results. Higher number of states shortened the text length. We expect more data to stabilize this issue.

6.3 Neural Models

We can see from the training and validation loss (Figure 5) that when training from scratch we were overfitting the training data, but that pretraining greatly improves the generalization by reducing the degree of overfitting, presumably by arming the model with general understanding of English character patterns before it tackles *Explain Like I'm Five*.

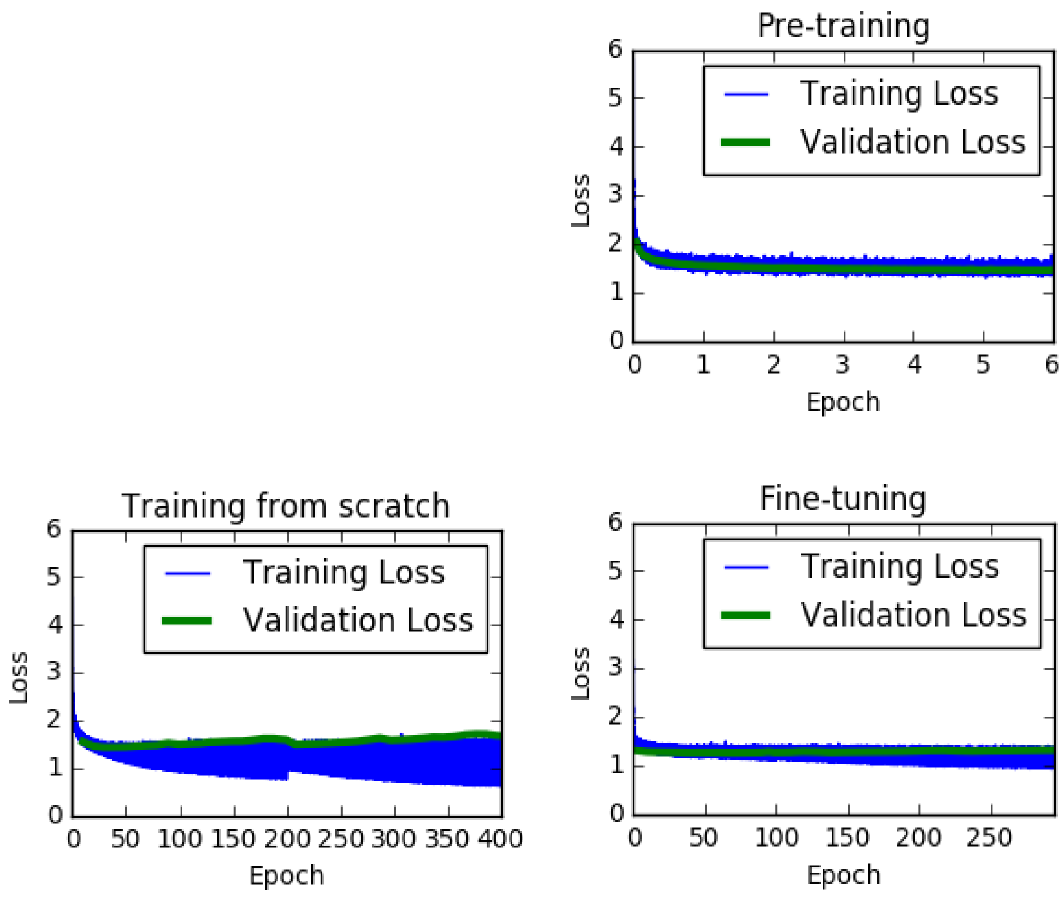


Figure 5: Training and validation loss of neural models

6.4 Model Comparison

Table 3 below compares the four models on the training requirements:

Model	Sampling time (s)	Training Time	Training Corpus size
n-gram	0.00162	Low	Moderate
Markovify	0.00351	Low	Moderate
Trained	0.15436	Medium	Large
Pretrained	0.22109	High	Moderate

Table 3: Model training dependencies

To illustrate the output of each models, Table 4 shows a sample of posts generated by the different models that is most representative of its outputs:

Model	Samples
n-gram	*eli5 : the pyramids ?* *eli5 : why do before i 'm seeing it would linguists find dead at all its diesel cars and is a 2 ?*
Markovify	*ELI5:Why are MMA fighter told not to refuel my sedan with the engine on?* *ELI5:If the Flu mutates each year, why does it move when you scratch it?*
Trained	*ELI5: What are the made from Encrything on white heat a voices?* *ELI5: How does excess pawelling water about a laltone so wholes/darks going to juption?*
Pretrained	*ELI5:How are computer hearing mental provinces in beginning of unperson supply?* *ELI5: Why do certain things are workouts?*

Table 4: Sample model output

Table 5 below gives the result of the human evaluation of the 100 sample outputs from the four models:

Model	Level	% Applicable	% Coherent	% Incoherent	Original ?
n-gram	word	11.0	15.0	74.0	Yes
Markovify	phrase	44.5	49.5	6.0	No
Trained	character	7.5	7	85.5	Yes
Pretrained	character	12.5	16.0	71.0	Yes

Table 5: Model comparison on performance

Table 6 below gives the inter rater reliability between the two human raters for the four models and also a combined Cohen's Kappa for the complete set of 400 samples. It is important to note that the overall agreement is not a plain average over the different models but rather a measurement of agreement over chance that measures the compatibility between the raters.

Model	Cohen's Kappa
n-gram	0.3893
Markovify	0.3532
Trained	0.1692
Pretrained	0.3303
Overall	0.4954 (moderate)

Table 6: Inter-rater reliability

7. Discussion

All models we tested were able to identify the regular structure of the *Explain Like I'm Five* subreddit, such as the prefix "ELI5:", and that posts typically begin with "How", "Why", and end with a question mark. However, the recurrent neural network models had some trouble learning to reliably produce recognizable words until we pre-trained them on a larger English language corpus (OANC[9]). With all the models, we struggled with having enough data due to the limitations of the Reddit API, so an obvious future direction would be to repeat the experiment with a larger body of data.

During our human evaluation, we saw that Markovify produced very high-quality output, but was able to do this by reusing large chunks of the input data; it didn't generate very new sentences, merely produced minimally-novel ones by swapping out a few elements (for example, "Why are some meats white while others are angry and mean?" was clearly borrowing from the input question "Why are some meats white while others are red? What's the difference?", only swapping out the end for "angry and mean?", which itself was borrowed from "Why some people are happy and loving drunks while others are angry and mean?").

Accurate tokenization is important to get good performance with an N gram model and *markovify*. Though we observed that higher capacity models were shortening the output, we could reason this was due to the small amount of training data for mid-ranged N values and state space sizes. In general, we observed during our human evaluation that longer sentences tend to lose coherence and were annotated as “incoherent”, though on average all models produced similar-sized outputs.

8. Future Work

There are many other relationships on Reddit that could be leveraged for better text generation. Many posts have descriptive bodies in addition to the post title, and a tree of comments rooted at the post. Although we restricted ourselves to generating post titles, it would be an interesting line of research to investigate models that try to parse the title, body, and higher comments in order to produce comments, or produce titles and bodies that are related.

The recurrent neural network models lend themselves to being combined with other neural models. One particularly promising area of expansion would be the image-based subreddits, which largely consist of images "captioned" by a post title. Combining convolutional neural networks with our recurrent neural networks, we should be able to improve greatly on the relatedness of the images and post titles as compared to the current Subreddit Simulator, which does not model the image at all.

The largest area for further improvement would be in evaluation; evaluating text generation is notoriously difficult. One thing we would like to investigate quantifying is originality; this would allow better separation of models that stick very close to the input, like Markovify, from models that really generate new sentences. Additionally, our human evaluation method's mediocre Cohen's Kappa score indicates that our scoring would benefit from more raters (for example with Amazon Mechanical Turk), or better rater training to make sure raters agree more on how to rate a particular output. Separately, we could also investigate model performance by using Reddit's inherent interactivity, posting model output to a Subreddit and gauging user response to posts. Finally, we could analyze the syntactic validity of output by using parse trees.

References

1. Bengio, Y, Simard, P, and Frasconi, P. (1994). "Learning long-term dependencies with gradient descent is difficult." *IEEE Transactions on Neural Networks*. 5(2):157–166.
2. Birch, C. (2016). "SubredditSimulator." *GitHub repository*. https://github.com/Deimos/SubredditSimulator
3. Bird, S, and Loper, E. 2004. "NLTK: The natural language toolkit." In *Proc. of 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*.
4. Boe, B & Dalool, E. (2016). "PRAW: Python Reddit API Wrapper." *GitHub repository*. https://github.com/praw-dev/praw
5. Crossley, SA, Kyle, K, and McNamara, DS. (in press). "The tool for the automatic analysis of text cohesion (TAACO): Automatic assessment of local, global, and text cohesion." *Behavior Research Methods.*
6. Glorot, X, & Bengio, Y. (2010). "Understanding the difficulty of training deep feedforward neural networks." *Aistats*. 9:249–256.
7. Graesser, AC, McNamara, DS, Louwerse, MM, & Cai, Z. "Coh-Metrix: Analysis of text on cohesion and language." *Behavior Research Methods, Instruments & Computers* (2004).
8. Hochreiter, S, and Schmidhuber, J. (1997). "Long short-term memory." *Neural Computation*. 9(8):1735–1780. ISSN 0899-7667
9. Ide, N, and Suderman, K. (2007). "The open american national corpus (oanc)." http://www.anc.org/data/oanc/
10. Johnson, JC. (2016). "Efficient, reusable RNNs and LSTMs for torch." *GitHub repository*. https://github.com/jcjohnson/torch-rnn
11. Mahoney, M. (2005). "Adaptive weighing of context models for lossless data compression." *Florida Inst. Technol., Melbourne, FL, Tech. Rep*. CS-2005-16.
12. Martens, J. (2010). "Deep learning via Hessian-free optimization." *Proceedings of the 27th International Conference on Machine Learning (ICML)*.
13. Reddit community. (2016). "Explain Like I'm Five." *Reddit*. https://www.reddit.com/r/explainlikeimfive/
14. Rissanen, J, and Langdon, GG. (1979). "Arithmetic coding." *IBM Journal of Research and Development*. 23(2):149–162.

15. Siegel, S, Castellan Jr., NJ. (1988). "Nonparametric Statistics for The Behavioral Sciences." *McGraw-Hill Book Company, New York.* ISBN-13: 978-0070573574.

16. Singer-Vine, J. (2016). "Markovify: A simple, extensible Markov chain generator." *GitHub repository*. https://github.com/jsvine/markovify

17. Sutskever, I, Martens, J, Hinton, GE. (2011). "Generating text with recurrent neural networks." *Proceedings of the 28th International Conference on Machine Learning*.