Stanford University

Exploring ways of Generating Free Text Unconditionally and Conditionally

Cécile Logé
Department of Computer Science - Stanford University
ceciloge@stanford.edu

Abstract

Generating free text remains one of the biggest challenges of AI for two reasons: (1) very often, grammar and consistency quickly become a trade-off to creativity, and (2) it remains difficult to have any control over the output once the model starts generating. In this study, we use several approaches (RNN, Gated CNN, Attention, Encoder-Decoder) to generate short movie plot snippets and explore ways to increase control by conditioning the outputs on a specific genre such as comedy, horror, thriller. Our final model, the optimized Conditional Encoder-Decoder (described in section 5.2) puts the idea of a trade-off to the test by obtaining the best consistency score while maintaining grammar and creativity levels on par with other high-performing models, and allowing for controlled text generation based on a chosen genre.

1 Introduction

Text generation is a common task in many NLP domains, ranging from text summarization to machine translation or even poetry generation. Our goal is to design an efficient language model to generate short text snippets in the form 2-3 sentences summarizing the plot of a movie (similar to a synopsis we would find on IMDb): ideally, the snippets would both make sense and be imaginative. In this study, we train several models with diverse architectures - RNNs with and without Attention, Gated CNN with Residual - on thousands of movie snippets of various lengths and genres - teaching them to take a few words as input and output a probability distribution for what the next word should be.
Among the various questions that this goal entails, one particular challenge stands out: how can we control the output of the model in a consistent manner (e.g. generating a movie synopsis for a particular genre) without restricting creativity, or renouncing structure and grammar? Inspired by previous work on this topic, we also explore ways to condition the outputs of our models on a specific genre (e.g. comedy, horror, action), including an Encoder-Decoder architecture that proves more efficient than expected.

2 Related work

Over the past few years, Deep Learning methods have surpassed previous probabilistic NLP techniques [1]: RNN architectures like LSTMs (Long Short-Term Memory) or GRUs (Gated Recurrent Units), and more recently Transformers are now considered state-of-the-art in Natural Language Processing. Promising results have often been the fruit of combining proven approaches from different fields in a creative way: for example, while the use of GANs in NLP is still considered a challenge - between the forced discretization of the output and problematic gradient computations - Lantao Yu et al. obtained encouraging results with SeqGAN [2] by mixing the concept of Adversarial Training with Reinforcement Learning and Policy Iteration. Dauphin et al. [3] also managed to outperform LSTMs in language modeling tasks by repurposing the Convolutional Neural Network architecture typically used in the domain of Computer Vision and adding Linear Gate Units between blocks. More recently, Keskar et al [5] introduced a Transformer model (CTRL) meant to replicate the degree of control available in image generation: the idea behind CTRL is to allow text generation to be probabilistically conditioned on a

specific control code (representing a theme, a style, a subgenre) after training the model on various sources of data including Wikipedia, Reddit, SQuAD. Inspired by these different approaches, we experiment with several models and architectures - some unconditional and some conditional - to output new artificial movie ideas.

3 Dataset, Features & Evaluation Metrics

Our dataset is made of 20,000+ datapoints consisting of a movie title, a movie overview (i.e. a snippet of text) and its genres. During the data cleaning process, we made sure to restrict the data to English language and Latin characters only, and removed datapoints whose overview was either too short (< 10 words) or too long (> 125 words). Additionally, we included <start> and <end> markers at the beginning and end of each snippet. Capitalization and punctuation, with the exception of the full stop " . " used as a sentence separator, were removed with the idea that both can then be added back manually at the end. From there, tokenization was conducted with Keras's Tokenizer and a vocabulary size of 10,000 words.
Although Text Generation modeling does not necessarily call for a test set, we put aside 400+ test data points for evaluation and experiments. To evaluate our results, we decided to use one 'objective' metric and four 'human' metrics:

- **BLEU Score**: a measure of closeness between the test string and the output string (using the first few words of the test string as a seed), as defined by Papineni et al [9].
- **Human Evaluation**: two people from different backgrounds took a survey to rate 5 outputs from each of our models - presented in random order - on the basis of **Grammar Quality**, **Story Consistency** and **Creativity**. For conditional models, we also add a fourth metric to measure the quality of the **Genre Attribution**.

4 Unconditional Models

4.1 Baseline GRU Model & Bidirectional Model

Our **Baseline model** borrows its simple RNN architecture from Keras's char-level language model designed by F. Chollet [6]. As shown in Figure 1, it consists of one Embedding layer, followed by two stacked GRU layers with 256 hidden units and tanh activation. From there, only the last state of the second GRU layer is kept and used as input in the second part of the model: two Dense layers (512 followed by 2560) with ReLU activation and a final Softmax layer for classification (10,000 x 1). This architecture will be the foundation of most following models.
With the idea to strengthen the model's capacity to retain long-term dependencies, we also experiment with bidirectionality and make both GRU layers from our Baseline model bidirectional. Just like the original model, this **Bidirectional model** only takes past tokens into account, but this time it also "reads" them in reverse order so as to become more aware of words that are further away in the sequence. Both left-to-right and right-to-left outputs are then concatenated and fed into the Dense/Softmax layers.

4.2 Attention Model with Concatenation

Building on our previous idea, we wanted to find a way to allow our model to choose which parts of the input sequence to focus on when making its prediction. In this **Attention model**, described in Figure 1, we start out with the same Embedding & GRU layers as in our Baseline model, but this time we keep track of the sequential outputs of each layer and concatenate them all before passing them into an Attention block.
In the Attention block, for $V^{(i)<j>}$, the concatenated vector for datapoint (i) and timestep j (out of $t-1$ timesteps in the sequence, as we're predicting the tth word), we compute the scalar $\alpha^{(i)<j>}$ as:

$$\alpha^{(i)<j>} = \frac{\exp(WV^{(i)<j>})}{\sum_{k=1}^{t-1} \exp(WV^{(i)<k>})}$$

$$\text{AttentionOutput}^{(i)} = \sum_{j=1}^{t-1} \alpha^{(i)<j>} V^{(i)<j>}$$

The goal is to compute an Attention-like score for each timestep (with W to be learnt) and output a weighted average of the sequence (i.e. a single vector representing the full sequence but with each token weighted based on its relevance). This single vector is then fed into the Dense/Softmax layers to predict the next word t.

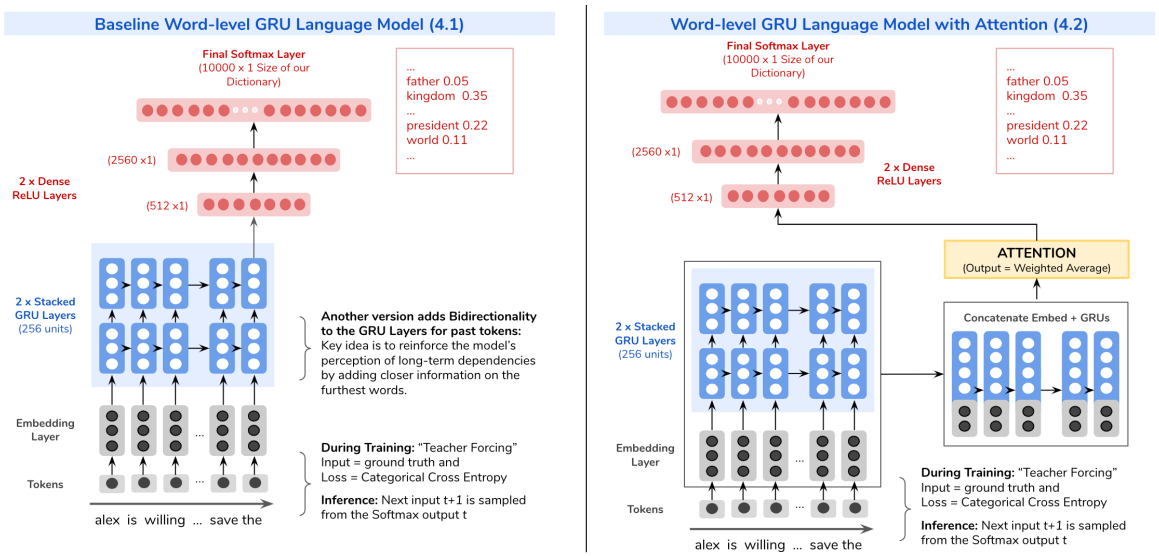

Figure 1: Left: Baseline Model / Right: GRU Model with Attention

4.3 Gated CNN Model with Residual

So as not to restrict ourselves to RNN architectures, we build on the work of Dauphin et al [3] to design a **Gated CNN model**: starting from a simple 1D Convolutional architecture, we incorporate Gated Linear Units into each Convolutional block as a way to adapt the filters to sequential data and give the model an opportunity to select the most relevant words/features to predict the next word. Described in Figure 2, this model consists of three Gated blocks, followed by two Dense layers (512 followed by 2560) with ReLU activation and a final Softmax layer for classification (10,000 x 1). Each gated block is structured as: two 1D Convolution layers A and B with 128 4x4 filters and 'causal' padding (meaning we pad only on the left so as to avoid sequential leakage and make sure only past context is used by the filter), followed by Batch Normalization layers for each convolution and completed with a Gated Linear Unit that combines both linear and non-linear elements and works as an activation function:

$$C_1 = \text{GLU}(A_1, B_1) = A_1 \odot \sigma(B_1)$$
$$C_2 = \text{GLU}(A_2, B_2) = A_2 \odot \sigma(B_2)$$
$$C_3 = \text{GLU}(A_3, B_3 + C_1) = A_3 \odot \sigma(B_3 + C_1)$$

The result C_j is the Hadamard product between the output A_j from the first 1D Convolution layer (after Batch Normalization) and the element-wise sigmoid of the output B_j from the second 1D Convolution layer (again after Batch Normalization) - with a skip-connection between C_1 and C_3. We experimented with and without the skip-connection and saw an increased speed and stability during training with the skip-connection.

4.4 Fine-Tuned GPT-2

We used Transfer Learning from GPT-2 [4] to build a new model for our specific task: we downloaded the "Small" 128m model and fine-tuned the weights by retraining on our movie snippets for a 800 epochs. This model will serve as a solid reference during evaluation.

5 Conditional Models

In this section, each datapoint in our training set is now attached to a (20x1) binary vector that gives information about the genres of the movie. These vectors are not unique: each movie can have up to 20 genres (although in reality, the maximum here turns out to be 8) and several movies - provided they have the same genres - will end up with the same vector.

5.1 Adding a Genre Feature to 4.2

A simple idea is to add a separate genre input to the Attention model we built in 4.2. In this new model, the sequence input follows the same path as before, while the genre input first goes through a ReLU Dense layer. Both paths cross after the Attention step via a Concatenation Layer: essentially, this means that the input to the final Softmax Layers will not only contain information about the past tokens (just like in 4.2), but also about the genres. The goal is to force the model to make more targeted predictions by feeding it more information in the last steps.

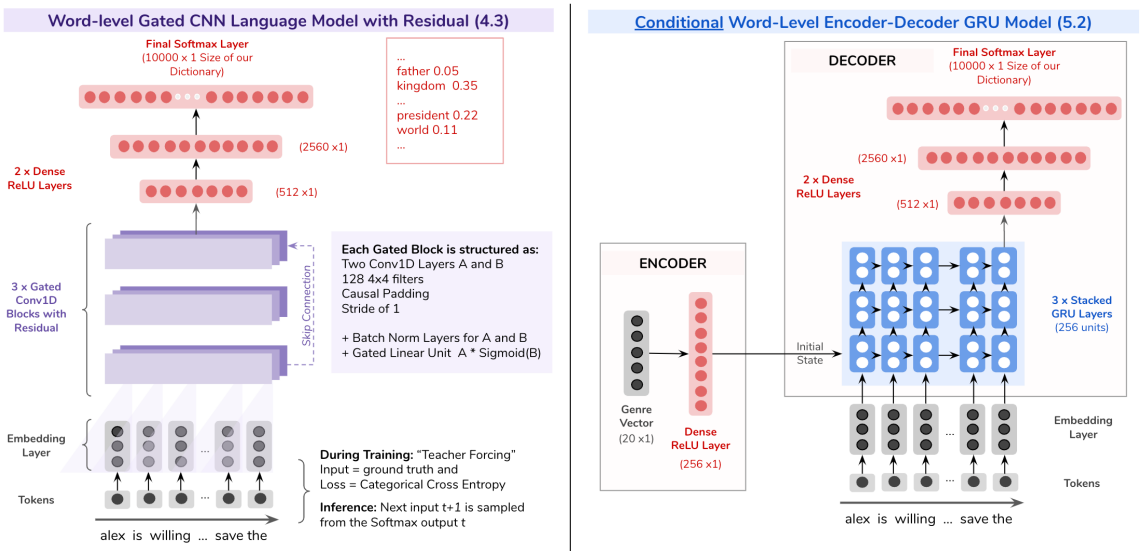

Figure 2: Left: Gated CNN Model / Right: Encoder-Decoder

5.2 Encoder-Decoder Model

Inspired by Machine Translation [7], we build this **Encoder-Decoder model** with the idea of increasing the genre's space of influence: instead of intervening only at the end of the model, just before the Dense layers like in 5.1, the genre now comes in much earlier to condition the whole model - including the RNN weights.
As shown in Figure 2, the Encoder block first takes the genre vector as input - into a simple Dense layer with 256 hidden units - and outputs an encoding that will then be used as the initial state of the Decoder block - along with the Decoder input (i.e the usual $t-1$ word sequence used to predict the tth word). The Decoder consists of three stacked GRU layers with 256 hidden units and tanh activation, followed by two Dense layers (512 then 2560) with ReLU activation and a final Softmax layer for classification (10,000 x 1).

6 Experiments, Results & Discussion

6.1 Loss, Training, Optimization & Inference

Loss is the typical Categorical Cross Entropy loss associated with Softmax: for each datapoint (i), the loss is computed several times as we loop over all the $T^{(i)}$ words in the snippet following the seed. Here $y_w^{(i)<t>}$ represents the ground truth for the word t in datapoint (i) with regards to word w in the dictionary (1 if it's word w, 0 otherwise), and $\hat{y}_w^{(i)<t>}$ is the prediction:

$$L(y^{(i)}, \hat{y}^{(i)}) = -\frac{1}{T^{(i)}} \sum_t \sum_w y_w^{(i)<t>} \log(\hat{y}_w^{(i)<t>})$$

Training was conducted in a "Teacher Forcing" fashion, with the last layer meant to predict the next word t conditioned on the previous words and looping over each word for each datapoint in the training set. For simplicity and a less memory-expensive implementation, we use a fixed historical window of 10 words.

For optimization, we used Adamax, a variant of Adam based on the infinity norm, as its sparse implementation is known to be superior in models with embeddings [8]. For each model, we made sure to adjust the **learning rate** step by step in a discrete staircase way, closely following the model's progress and ranging from 0.04 to 0.0001. After hitting a few loss plateaus during training, and losing sensibility from the learning rate, we also experimented with **initialization**: switching from Xavier Uniform (default in Tensorflow) to Xavier Normal for the GRU layers, and from He Uniform (default in Tensorflow) to He Normal for the Dense ReLU layers made training not only much faster but also more sustainable and ultimately much more efficient.
At inference time, we produce text by running the model on a loop in a "One to Many" fashion: we use a seed of ten words, sample the next word from a multinomial distribution over the model's Softmax prediction vector and add it to the end of our string. We then rerun the model on this updated string and go on until we've hit the <end> token. When scoring our models with BLEU however, we use Greedy Sampling to remove any random noise in our results.

6.2 Results & Discussion

	BLEU Score	Grammar	Consistency	Creativity	Genre
Baseline GRU RNN (4.1)	6.4%	0.40	0.28	0.47	-
Bidirectional GRU RNN (4.1)	6.6%	**0.66**	**0.48**	0.40	-
Attention GRU RNN (4.2)	6.5%	0.41	0.31	**0.64**	-
Gated CNN (4.3)	7.1%	0.43	0.37	0.39	-
Attention with Condition (5.1)	6.9%	0.43	0.31	**0.58**	0.33
Encoder-Decoder (5.2)	6.3%	**0.61**	**0.61**	0.57	**0.65**
Finetuned GPT-2 (4.4)	-	**0.80**	**0.72**	**0.57**	-

Figure 3: Result Overview / Objective Metric (BLEU) & Human Evaluation Metrics (Scores out of 1)

As seen in Figure 3, our **Bidirectional model** obtains the best performance in terms of grammar and consistency out of all unconditional models, improving significantly over the baseline except for creativity, while our **Attention model** led to the highest creativity score but progressed only very slightly in grammar and consistency. Interestingly, in this case, grammar and consistency turned out to be highly related to the average length of the output: adding Bidirectionality led to shorter outputs on average, often limited to 3-5 words after the seed until hitting the <end> token - which makes it understandably easier to hit higher consistency and better grammar but also explains the lower creativity score. This would seem to confirm the idea of a trade-off between grammar/consistency and creativity.
However, this idea was put to the test when our **Encoder-Decoder model** managed to improve all three scores over the baseline, even achieving the highest score in consistency while maintaining high creativity comparable to our fine-tuned GPT-2 reference. While the genre condition itself was only mildly successful at 0.65, it allowed us to confirm that adding the condition too late in the model - like in 5.1 - would not be efficient enough for output control. Additionally, as seen in both conditional models, adding a condition to the model seemingly helps improving consistency and grammar by helping the model stay focused and targeted.
In all models, one common problem stood out particularly: the risk of entering a repetition loop as soon as a rare word is outputted. While this is a well-known problem [10] in language modeling, it was quite interesting to witness it in practice.

7 Conclusion & Future Work

Overall, we were able to achieve encouraging results and interestingly, our optimized Conditional Encoder-Decoder model showed strong performance in terms of consistency and grammar, even surpassing other unconditional models without trading off creativity. This seems to indicate that applying more control over the output does not necessarily imply restricting creativity and may even lead to higher consistency and better grammar. While it is still far from the performance of our Finetuned GPT-2 model when it comes to outputting consistent stories in a correct grammar, this leads to interesting directions for future research, with a solid potential to improve our results. In particular, if we had more computer resources and time, we could focus on:

- Building an Hybrid model to leverage the qualities of each of our approaches, e.g. combining Attention and Bidirectionality in our Encoder-Decoder model,
- Exploring Transformer architectures and building on the work of Keskar et al [5] on controllable output.

8 Contributions

This project is 100% the work of Cécile Logé. Code is available at: github.com/cecileloge/Free-Text-Generation.

References

[1] Neural Text Generation: A Practical Guide, Ziang Xie, 2017, arXiv 1711.09534

[2] SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient, Lantao Yu, Weinan Zhang, Jun Wang and Yong Yu, 2017, arXiv 1609.05473

[3] Language Modeling with Gated Convolutional Networks, Yann N. Dauphin, Angela Fan, Michael Auli and David Grangier, 2017, arXiv 1612.08083

[4] GPT-2 Language Models are Unsupervised Multitask Learners, Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei and Ilya Sutskever, 2019, OpenAI blog

[5] CTRL: A Conditional Transformer Language Model for Controllable Generation, Nitish Shirish Keskar, Bryan McCann, Lav R. Varshney, Caiming Xiong and Richard Socher, 2019, arXiv 1909.05858

[6] Char-level text generation with LSTM by keras.io

[7] Char-level recurrent seq-to-seq model by keras.io

[8] Adam: A Method for Stochastic Optimization, Diederik P. Kingma and Jimmy Ba, 2017, arXiv 1412.6980

[9] BLEU: a Method for Automatic Evaluation of Machine Translation, Kishore Papineni, Salim Roukos, Todd Ward and Wei-Jing Zhu, 2002, ACL July02 pp. 311-318

[10] Cutting-off redundant repeating generations for neural abstractive summarization, Suzuki, Jun, and Masaaki Nagata, 2016, arXiv 1701.00138

[11] Libraries: Numpy, Pandas, Keras, TensorFlow, NLTK, scikit-learn

See Appendix in next page for sample outputs.

Appendix: Selected Sample Outputs

SELECTED SAMPLES

(Capitalization added back manually / <start> and <end> tokens removed)

Baseline Model 4.1

[SEED: Alex is willing to accept a new mission to save] his life and save a group of stranded as they attempt to steal the weapons and avoid lonely behavior. The film chronicles the struggles of plants and twisted men.
[SEED: A group of friends decides to go out for] their destination off to confront his own death legacy.

Bidirectional Model 4.1

[SEED: CS230 students met a year ago for] the first time.
[SEED: CS230 students discover a] dark conspiracy behind an explosion with a mysterious past despite a mad array of programming.
[SEED: Professor Andrew goes on an adventure to] the top of the ordinary.

Attention Model 4.2

[SEED: A group of friends decides to go out for] an undercover murder while a dogged war meteor crashes on the eve of graduation and South America.
[SEED: Professor Andrew goes on an adventure to] cool into the Ancient Land and take over the world while keeping or break old desires. They are over the other in order to escape their mundane family and hold in the woods.

Gated CNN Model 4.3

[SEED: CS230 students discover a] new statue of chance after Ellen learns she's harder than it seems...
[SEED: CS230 students met a year ago for] the fascinating trafficking spree.

Fine-tuned GPT-2 4.4

[SEED: CS230 students met a year ago for] their first time on a college campus, and what they learned is that life on campus is not as easy as it seems.
[SEED: A group of friends decides to go out for] a little fun and fight the giant monster.

Conditional Attention Model 5.1

Thriller/Crime: [SEED: A group of friends decides to go out for] their honeymoon. They find themselves pulled into knife city situation.
Thriller/Action: [SEED: Alex finds out that Judy is] mischievous wolf whose wife Joanna is very domineering.

Conditional Encoder-Decoder 5.2

Thriller/Crime: [SEED: A group of friends decides to go out for] another day when they are framed for a kidnapping inmate and his little daughter.
Comedy/Romance: [SEED: Professor Andrew goes on a mission to] discover the identity of a nasty boyfriend.
Mystery/Drama: [SEED: Alex is pretty rich and famous but] she discovers a mysterious stash of violent diamonds.

Figure 4: Selected outputs / Capitalization added back manually