Text generation and Style Transfer
Data Vaders - Group 17

Soumye Singhal
150728
soumye@iitk.ac.in

K. Siddarth
150312
siddarth@iitk.ac.in

Prakhar Agarwal
150499
pkhrag@iitk.ac.in

Abhibhav Garg
150010
abhibhav@iitk.ac.in

Nishit Asnani
Mentor

1 Introduction and Motivation

Our project has two parts.
The first part deals with **Text Generation**. Text generation is the fundamental task in Natural Language Processing which aims to produce a natural language text in order to meet specified communicative goals. It takes the non-linguistic representation of information as input and outputs text, documents, stories etc. It has a diverse set of applications ranging from image captioning to text summarization. We aim to use current state of the art Deep Learning techniques to capture the artistic style of an author and then generate text in that same style. Formally, given a starting sentence and a target author (say JK Rowling or Shakespeare), we use text generation methods to complete a paragraph in the style of that author. We compare the performance of RNN's and LSTM's. We also see how the performance changes with the change in the architecture.
The second part deals with **Style Transfer**. We want to use deep learning based seq2seq models to change the style of a given content from that of the original author to that of a new target author. More formally, given some text of any author, we want to convert it to the style of another, while still preserving the meaning of the text. We experiment with our original idea and analyze its performance.
We think that pursuing this task will bring us closer to an algorithmic understanding of how our brains are able to create and perceive artistic literature and how it captures style.

2 Related/Previous Work

Our work on text generation was inspired by Karpathy's[5] work on text generation using character level models. In his article, he goes over the effectiveness of recurrent models in text generation.

Our work on style transfer of text was inspired by Gatys et. al.'s[3] work on style transfer of painters in vision.
There has not been much work on transferring the style of the text of one author to another. Hu et. al.[4] did controlled text generation using Variational Autoencoders[6]. But their work dealt with generating any paraphrase of a given input. What we want to do is create a specific type of paraphrase, not just any paraphrase. In that respect, this problem is similar to machine translation, where we want to translate text from a given input language to a different output language. Sequence to sequence encoder-decoder models have proven to be very successful in that regard. Bahdanau et. al. [2] gave state-of-the-art results in machine translation in 2014.

3 Models

3.1 Text generation

Our text generation model is based on simple Recurrent Neural Network(RNN) based language model trained on the text of the target author. The RNN learns to predict the next word given the previous context. Since it's trained on the target authors corpus, when we feed in the context of our input string it starts to predict the next word in the style of that author. We then keep on picking this predicted word to generate and complete the text in the style of that author.

In the model described in this Figure, h_i denote hidden state of the RNN. The network outputs a

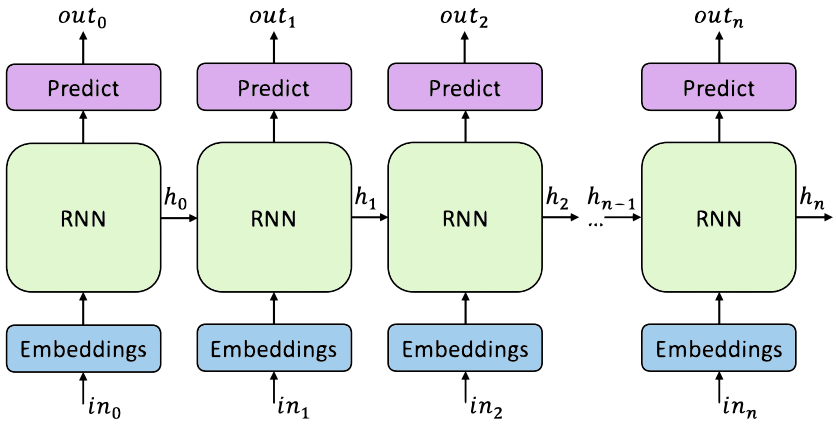
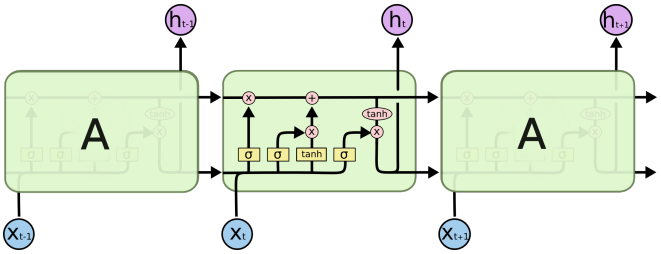


Figure 1: Recurrent Neural Network- At each timestamp, a character and some computation of last time-stamp is given as input to compute the next character.

distribution over vocabulary, from which we sample out the output tokens out_i.

- w_i - input tokens of source text
- h_i - Encoder hidden states
- $P_{vocab} = softmax(Vh_i + b)$ is the distribution over vocabulary from which we sample out_i

The recurring unit can be either a vanilla RNN or an LSTM unit. LSTM's(Long Short Term Memory networks) improve on RNN's by maintaining a gating system, which mitigates the vanishing and exploding gradient problem experienced by vanilla RNNs. [1] We can implement this RNN network



in two ways. The inputs can be either characters or words. We correspondingly get either a character

[1] Image taken from `colah.github.io`

RNN or a word RNN. We experimented with both of them and compared their performance. We also experimented with different recurrent units like RNN's and LSTM's, varying stacked layers and varying hidden layer size. We present a comparison of results in the experiments section. We saw that our model performs best for word LSTM, with 512 size hidden layer and 3 stacked layers.

3.2 Style Transfer

Our model for style transfer from one author to another is based on the basic sequence to sequence encoder-decoder model with attention. These models are used to map an input sequence of words to another target sequence of words of possibly different vocabulary. They are used in tasks like machine translation, paraphrasing and text summarization.
Sequence to sequence models consist of an encoder and a decoder. The input is passed through the encoder to get a latent representation(called a thought vector). This is then passed through a decoder which decodes the latent representation to give the output.

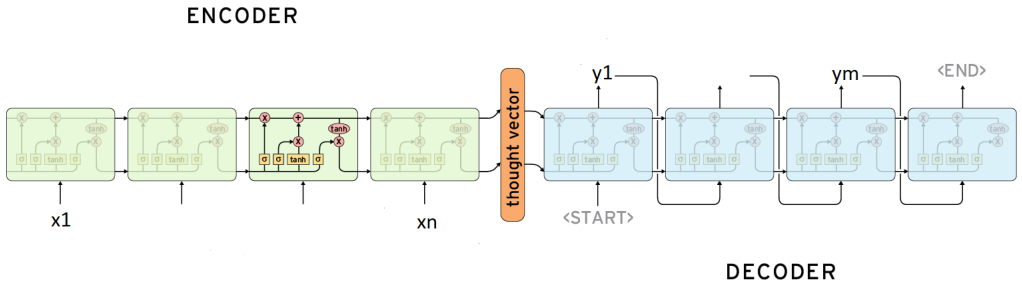


Figure 2: A seq2seq model

Attention Mechanism
The basic encoder-decoder model performs moderately well on very short sequences but it fails to scale up.

- The main bottleneck is the fixed sized encoding of the input string, which is the LSTM output of the last time-step. Due to it's fixed size it is not able to capture all the relevant information of the input sequence as the model sizes up.
- At each generation step, only a part of the input is relevant. So how does the model decide which part of the input encoding to focus on at each generation step?
- At each step, the decoder outputs hidden state h_i, from which we generate the output.

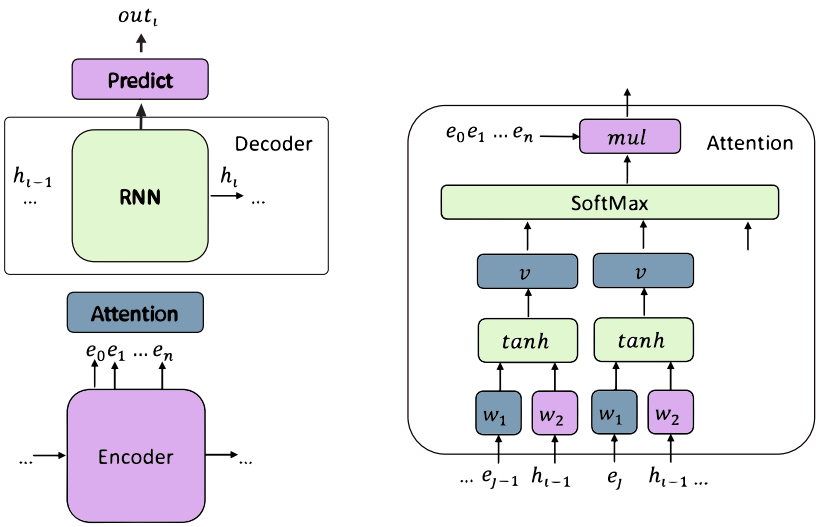
The solution to this problem is using the attention model[2]. This model calculates the importance of each input encoding for the current step by doing a similarity check between decoder output at this step and input encodings. Doing this for all of the input encodings and normalizing, we get an importance Vector. We then convert it to probabilities by passing through softmax. Then we form a context vector by multiplying with the encodings.

- $importance_{it} = V * tanh(e_i W_1 + h_t W_2 + b_{attn})$.
- Attention Distribution $a^t = softmax(importance_{it})$
- ContextVector $h_t^* = \sum_i e_i * a_i^t$

Context Vector is then fed into two layers to generate distribution over the vocabulary from which we sample.

- $P_{vocab}(w) = softmax(V'(V[h_t, h_t^*] + b) + b')$
- For the loss at time step t, $loss_t = -\log P(w_t^*)$, where w_t^* is the target word.

[2]Image stylized from https://talbaumel.github.io/attention/



- $LOSS = \frac{1}{T}\sum_{t=0}^{T} loss_t$ is the total loss across all time-steps.
- We then use the Back-propagation through time Algorithm to get the gradients and then apply any of the popular Gradient Descent algorithms to minimize the loss and learn good parameters.

Our Idea

We make a seq2seq encoder-decoder work as an auto-encoder first. That is given an input sentence, we train it to output the same sentence. We did this for `Agatha Christie`, `Shakespeare` and `Arthur Conan Doyle`. As these models can't handle multiple sentences well, we only train these on single sentence to single sentence. Once the seq2seq auto-encoder is trained, we take the input from Author 1(say `Arthur Conan Doyle`), and encode it using the encoder trained on Author 1. We then take the thought vector generated by it and pass it through the decoder trained on Author 2(say `Agatha Christie`).

Why we think it should it work?

An autoencoder seq2seq network would first learn a good encoding of the sentence and then using that it learns to regenerate the sentence. Therefore, while training on an author the network needs to encode the sequence in general way such that it's possible to recreate the sequence from that encoding. The decoder which is working with the encoder also has to learn to decode that sequence and learns the structure of the sentence that doesn't vary from sentence to sentence. Therefore, it makes sense for the model to encode only the content part of the sentence since the style remains constant for a given author(so the encoder would prefer to encode the more informative part, i.e. the content). And then the decoder can learn to recreate the given encoded content in the style of that particular author. So if we have two auto-encoder seq2seq models trained on two different authors, we can club the encoder of one and the decoder of another to get a model that does style transfer from one author to another

In our **final model**, we used a bi-directional LSTM for encoder and simple LSTM for decoder. We used hidden state size of 1024 and 2 stacked layers of LSTMs. The embedding size for the words was 500. We used beam search with a beam width of 5 to sample, and the maximum decode length was 300.

4 Dataset

For the text generation part of our work, we used the TinyShakespeare dataset. It contains the concatenation of the works of **Shakespeare**. There are 40k sentences in the dataset.

For the style transfer part, we took our data from Project Gutenberg.[1] We collected the works of **Sir Arthur Conan Doyle** and **Agatha Christie** from Project Gutenberg to make a tiny dataset for each author, each of which had roughly 50k sentences.

5 Tools and Libraries used

We used Tensorflow to code our project. We used Google's NMT code [7] as a boilerplate for seq2seq models. We coded the character/word RNNs ourselves from scratch.

6 Experiments

Hyperparameter Tuning

We did hyperparameter tuning(depth and size of the model) manually. We tested it for a few cases to find the optimal value.

Character vs Word level RNN

First, we wanted to compare the effectiveness of character level language models versus word level models. We trained two recurrent neural net models, one on trained on characters and one on words. For both, we used 3-layer stacked LSTMs with 512 hidden nodes. These were the results we got:

Word Level

> King VI: First Citizen: And will will tell you, I have not I is to be content; it are not that is a more than all the writing.
> DUKE OF YORK: My lord, I am a bond, and we is the writing.
> DUKE OF YORK: What is the writing.

Character Level

> **KINGequeses**, wifely A mighty **vanagy** died, and is it **sotis** being note but by flatter, which, I rather be! Hear over-blown swifled by

We can see that the character level model performs admirably; it is able to learn the syntactic structure of the text, and even some commonly used words like *being*, *which*, *be*, etc. But it still conjures up new words, like *vanagy*. This is not a problem with the word level model.

Depth

Next, we wanted to find the optimal depth for our model. Curiously, we found that on using a vanilla RNN stacked with 3 layers and 512 hidden nodes performed visibly worse than a 2 layer stacked RNN with the same hidden node width.

2 layer vanilla RNN

> KING RICHARD III: Ay, if you know the general is not so far with me.
> QUEEN ELIZABETH: My lord, I will not not a man of such good Than not to see him in the Duke of York.
> KING RICHARD III: Ay, but you will not be a traitor to the people, And yet thou art a soldier, and that is not so much with me for his eye.

3 layer vanilla RNN

> KING of

This problem does not arise when we use LSTMs instead of RNNs

3 layer LSTM

King VI: And will will tell you, I have not I is to be content; it are not that is a more than all the writing.
DUKE OF YORK: My lord, I am a bond, and we are the writing.
DUKE OF YORK: What is the writing

We believe the 3-layer LSTM outperforms the 3-layer RNN because gradients vanish much faster in vanilla RNN's than in LSTM's. During training of recurrent units, we want that the error to be back-propagated over all the time steps. With RNN's, the gradients very quickly tend to zero, meaning RNN's cannot learn long-term dependencies very well.
With the 2-layer RNN, the gradients have to be back-propagated over fewer layers compared to the 3-layer RNN, meaning fewer of the weights tend to zero. This allows the 2-layer RNN to perform better.

Style Transfer

We trained two seq2seq models, one on `Conan Doyle` and one on `Agatha Christie`. We then passed the input text(Sherlock Holmes) through the encoder of `Conan Doyle` to get the thought vector. This was then given as input to the decoder of `Agatha Christie`

To measure how well our encoder-decoder model does auto-encoding, we calculated the `BLEU` score. It essentially measures n-gram overlap between two texts, so higher `BLEU` score will mean better auto-encoding quality. We got a score of **55.13**%. This indicated that the encoder performs decently, but can be improved with more training data.

Results

Sherlock Holmes (Original)

Absolutely none. None. No sign of it? Come in ! said Holmes . Seven ! I answered . She will not sell . **And I.** My own seal . We have tried and failed . Stolen , then . I was mad - insane . To ruin me . We were both in the photograph .

Generated

Absolutely no. None . No sign of it ? Come in ! said . Lord ! I answered . She will not see . **And me.** My mother. We have come and rushed . Welcome , then . I was mad - . To me me . We both were in the photograph .

Sherlock Holmes (Original)

What do you make of that ? asked Holmes . **I am about to be married** . **I think that I had better go** , Holmes . My private note-paper . No legal papers or certificates ? I promise , said Holmes . I carefully examined the writing , and the paper upon which it was written .

Generated

What do you make of that ? asked asked. **I am going to be married** . **I think that I had really go** , I had My private private. No girl or or two ? I dare , said gruffly . I carefully the man , and the paper paper which it was written .

As we can see, the conversions are not very good. There are some examples(the highlighted ones) where we can see that the structure of the sentence changes, but the meaning is (mostly) maintained. Also, we have no way of evaluating how good our translated texts are, given that there is not much text on the same content written by two different authors.
We also tried to convert Sherlock Holmes to Shakespeare(since the language used is so different, it would be much easier to see if the style was transferred), but due to their vocabularies being so different, we got junk.

7 Future Work

There is a lot of scope for extending the style transfer model. There are several shortcomings of this model that can be addressed. We noted that the model has struggle with the words that aren't present in the target author's vocabulary. So instead of outputting a garbage work there, we can use a pointer-generator network which learns to copy a word from the input sequence and give that as the predicted output instead.

Another drawback of our model is that the model has to learn the word embeddings layer itself based on the training data. But since in our case the data is scarce for the model to learn very good word representations, we can user pre trained word-vectors like word2vec of GloVe in our model instead.

We can also use ideas from controlled generation of text to manually append a control variable to the encoding and then append a discriminator to the sequence so as to force the model to change the behavior of the output sequence based on the control variable we provide during testing. We can use this control variable to control various aspects of the style of an author.

References

[1] Project gutenberg. *http://www.gutenberg.org*.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2014.

[3] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. A neural algorithm of artistic style. *CoRR*, abs/1508.06576, 2015.

[4] Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P. Xing. Controllable text generation. *CoRR*, abs/1703.00955, 2017.

[5] Andrej Karpathy. The unreasonable effectiveness of recurrent neural networks. 2015.

[6] D. P Kingma and M. Welling. Auto-Encoding Variational Bayes. *ArXiv e-prints*, December 2013.

[7] Minh-Thang Luong, Eugene Brevdo, and Rui Zhao. Neural machine translation (seq2seq) tutorial. *https://github.com/tensorflow/nmt*, 2017.