Imperial College London

Department of Computing

Chameleon Text: Exploring ways to increase variety in artificial data

Author:
Thien P. Nguyen

Supervisor:
Lucia Specia

Submitted in partial fulfillment of the requirements for the MSc degree in Computing (Artificial Intelligence) of Imperial College London

May 2019

Chapter 1

Literature Survey

1.1 Background

Text generation is a a type of Language Modelling problem, which in itself, is one of the core natural language processing problems, and it used in a variety of contempoary applications, ranging from machine translation, to email response generation, to document summarisation.
In our particular case we imagine a scenario where a client requests the use of our dataset. We would permit them access to said data, but it would reveal the identities of the users in the data. The objective is to have some alternative dataset to ours such that value can be deduced from the data, but the privacy of the users in our original dataset is maintained.
This can be accomplished by creating a language model that would increase the lexical variety of our original dataset. This language model would train on our original dataset, and would produce data that is semantically and lexically similar to our original data, but diverges enough such that it could potentially be seen as an entirely new and independent dataset. This new dataset can be assigned to other clients.
The premise of this literature survey is to describe the relevant components necessary to construct our solution, and to also discuss alternative approaches to the problem.

1.2 Language Modelling

Language modelling is the task of predicting a word w_i in a text w given some sequence of previous words $(w_1, w_2, ..., w_{i-1})$. More formally, Dyer (2017) describes an unconditional language model as assigning a probability to a sequence of words, $w = (w_1, w_2, ..., w_{i-1})$. This probability can be decomposed using the chain rule:

$$p(w) = p(w_1) \times p(w_2|w_1) \times p(w_3|w_1, w_2) \times ... \times p(w_i|w_1, w_2, ..., w_{i-1}) \tag{1.1}$$

$$p(w) = \prod_{t=1}^{|w|} p(w_t|w_1, ..., w_{t-1}) \tag{1.2}$$

Traditionally, assigning words to probabilities may conflate syntactically dubious sentences but it remains to be a useful method for representing texts.
In particular, we are more interested in conditional language modelling. This slightly differs from the definition described above - A conditional language model assigns probabilities to sequences of words, $w = (w_1, w_2, ..., w_{i-1})$, given a conditioning variable, x.

$$p(w|x) = \prod_{t=1}^{|w|} p(w_t|x, w_1, ..., w_{t-1}) \tag{1.3}$$

Le (2018) says the most fundamental language model is the n-gram model. An n-gram is a chunk of n consecutive words. For instance, given the sentence ”the quick brown fox . . . ”, the respective n-grams are:

- unigrams: ”the”, ”quick”, ”brown”, ”fox”
- bigrams: ”the quick”, ”quick brown”, ”brown fox”
- trigrams: ”the quick brown”, ”quick brown fox”
- 4-grams: ”the quick brown fox”

The intuition of n-grams was that statistical inference can be applied on the frequency and distribution of of the n-grams, which could be used to predict the next word. However, sparsity is not captured.
Modern language models revolve around the use of neural networks, which was introduced by Bengio et al. (2001), with a simple MLP that encoded words. The use of neural networks in language modelling is often called Neural Language Modelling, of NLM for short.
Neural Networks are non-linear statistical models that generate complex relationships between input and output vectors. (This type of neural network architecture is commonly described as the multi-layer perceptron). Note that the input and output vectors are of a fixed dimension, which becomes a problem for our task at hand. Neural Networks evaluate an input using forward propagation, to produce an output. Traditionally, neural networks are trained to produce optimal outputs via the use of backpropagation.

1.3 Recurrent Neural Networks

Recurrent neural networks (RNNs) are a class of neural networks such that the outputs are not necessarily restricted and discrete (as opposed to the MLP). RNNs operate over a sequence of variable-length vectors, and produces an output of similarly variable-length vectors. This circumvents a problem introduced with using an MLP, where sentences are not typically fixed length. RNNs can XXXXXXXXXXXXXXX.
The architecture of RNNs make it favourable in NLP related problems as words in sentences are typically conditioned on the previous words. Given a sequence of

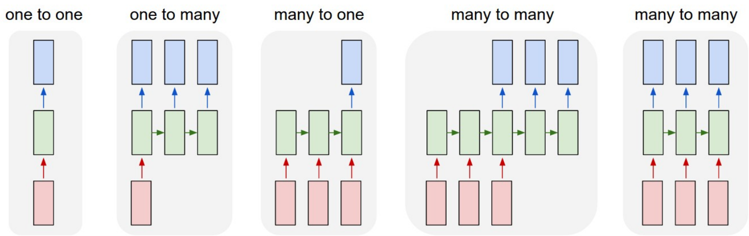

Figure 1.1: Rectangles represent vectors, with red being the input, blue the output, and green representing the state of the RNNs. Arrows represent functions. From left to right: (1) an MLP. (2,3,4,5) are examples of different styles of recurrent neural networks, describing the different types of input and output combinations. (Diagram from Karpathy (2015))

inputs $(x_1, ..., x_M)$, a standard RNN computes a sequence of outputs $(y_1, ..., y_N)$ by performing forward-propagation in a similar to fashion to MLPs, but outputs are chained together as an additional input to other neural networks. This can be visualised best in Figure 1.1. RNNs are trained using backpropagation-through-time.

RNNs showed promise but there existed a multitude of problems. Firstly, It became aparrent that it was very difficult for RNNs to leverage relationships between potentially relevant inputs and outputs - there isn't necessarily a clear indicator in the architecture that would facilitate this feature. This is described as a long range dependency problem. Furthermore, they were especially impractical due to the vanishing and exploding gradient problems having a larger effect on training. This is by design, as the architecture of the network does not provide the affordance to represent relationships between arbitary cells in the network.

1.3.1 LSTM

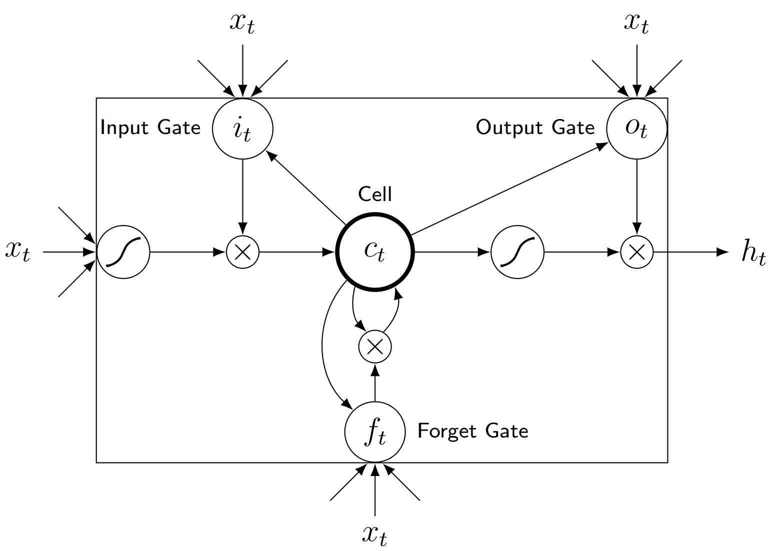

Figure 1.2: A diagram of the LSTM model.

LSTM (Long Short Term Memory) is a type of RNN unit that attempts to retain information based on the previous inputs through the introduction of gated architectures.

1.3.2 Gated Recurrent Units

Introduced by Cho et al. (2014), GRUs are used to solve the common issue with LSTMs where the training time was relatively slow due to the number of derivatives necessary to compute. Within the GRU architecture, a feature to retain the previous weights remain, but there exists an direct path to the input data from the output, allowing a reduction in training time. They are unable to clearly distinguish between the performance of the two gated units they tested. However, research from Chung et al. (2014) that GRUs were found to perform better than LSTMs on smaller datasets.

1.4 Autoencoders

Autoencoders are a specialised form of MLPs where the model attempts to recreate the inputs on the output. Autoencoders typically have a neural network layer in the model where its dimension is smaller than the input space, therefore representing a dimensionality reduction in the data. Autoencoders are composed of two different principal components, an encoder network α and a decoder network β, such that $\alpha : X \to F$ and $\beta : F \to X$. Measuring the success of the reconstruction is deduced by a reconstruction loss formula. This reconstruction loss comapres the output of the decoder and compares it against the input of the encoder. The two networks are trained together in a manner that allows them to preserve the input as much as possible.
Autoencoders are popularised through their use in Machine Translation, Word Embeddings, and document clustering.

1.5 Related Work

1.5.1 Seq2Seq

Seq2Seq, introduced by Sutskever et al. (2014) is a modern interpretation of the autoencoder model,
Traditionally, Seq2seq would be the most common method of tackling this particular problem, but it also presents problems that make it suboptimal. Firstly, it's discrete nature suggests that it is prone to noise in a similar fashion to how linear regression is not necessarily optimal as opposed to

1.5.2 Seq2Seq with Attention

Further work by Bahdanau et al. (2014) improved the original seq2seq model by providing an attention mechanism. The attention mechanism looks at all of the in-

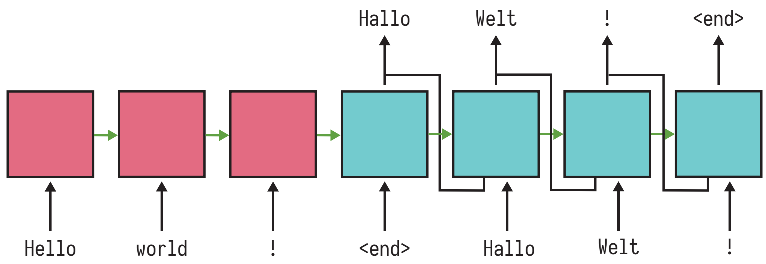

Figure 1.3: An abstracted model of the seq2seq architecture, where the encoder (pink) takes in the input sequence, and the decoder (blue) shows the output sequence.

puts from the hidden states of the encoders so far. This allows the decoder network to "attend" to different parts of the source input at each step of the output generation. This circumvents the need to encode the full source input into a fixed-length vector, helping it deal with long-range dependency problems.
Furthermore, this attention mechanism allows the model to learn what to attend to based on the input sequence and what it has so far, represented through a weighted combination of the two. Bahdanau et al. (2014) have found this to work especially well in machine translation problems where languages which are relatively well aligned (such as English and German) - The decoder is most likely able to choose to attend to the response generation sequentially, such that the first output from the decoder can be generated based on the properties of the first input of the encoder and so on.

1.5.3 Attention Mechanism

Attention mechanisms fhlajsdfhaljsdfhasldkjh

1.5.4 Transformers

Vaswani et al. (2017) introduced the idea that it is possible to avoid the use of RNNs altogether and focus on leveraging the attention mechanism introduced in seq2seq. The resulting network architecture utilises stacked layers of residual networks for the encoder and the decoder.

1.6 Variational Autoencoders

Our proposed solution revolves around the use of Variational Autoencoders. Variational Autoencoders (VAEs) introduce a constraint on the encoding network that forces the model to generate latent vectors that roughly follow a gaussian distribution, as opposed to creating a fixed latent vector. Consequently, VAEs introduce two extra vectors, a mean vector and a standard distribution vector, which is fed from the encoder. A sample is taken from the distribution and that is then fed into the decoder.

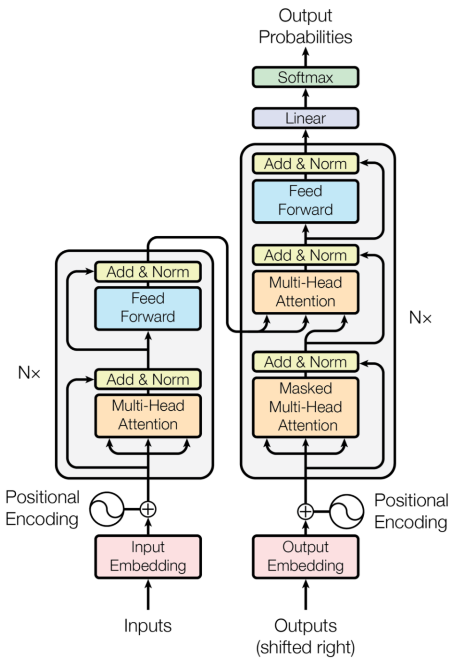

Figure 1.4: The transformer model architecture.Vaswani et al. (2017)

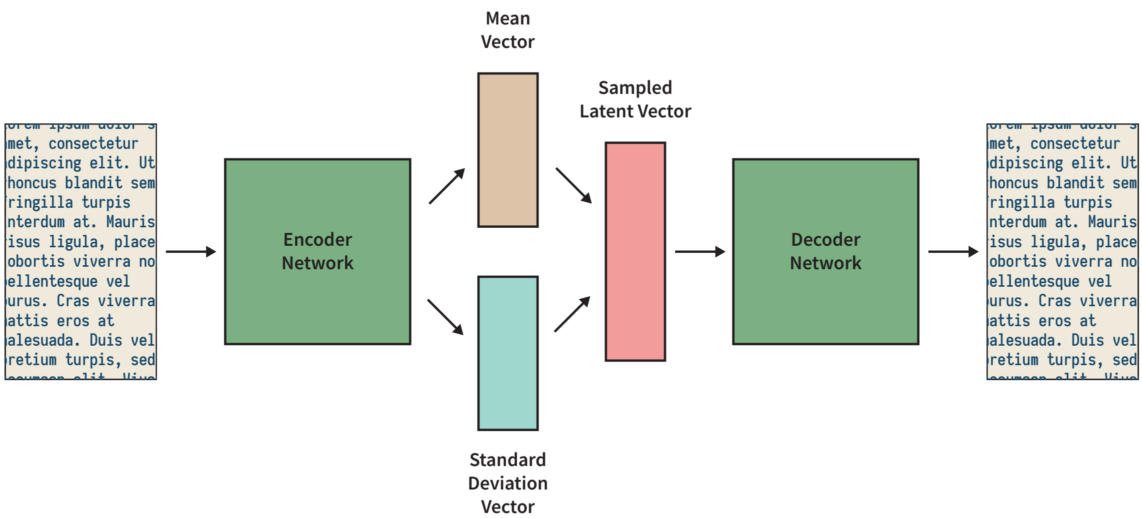

Figure 1.5: A simplified model architecture for a variational autoencoder, which takes as input some text, and it's predicted output being the same text as the input.

Note that the decoder receives samples from a non-standard normal distribution produced by the encoder. The average of the samples of the different distributions should approximate to a standard normal.

Due to the stochastic nature of the network, we use a reconstruction loss that involves an expection of the output; but we also use the KL divergence, which measures the relative difference of two probability distributions. In this particular case, we will be comparing the distribution of the decoder outputs against a standard gaussian $\mathcal{N}(0, 1)$).

$$\mathcal{L}(\theta, \phi, x, z) = \mathbb{E}_{q\phi(z|x)}[log\, p_\theta(x|z)] - D_{KL}(q_\phi(z|x)\,||\,p(z))$$

$$D_{KL}(P||Q) = \sum_{x \subset X} P(x) \cdot log(\frac{P(x)}{Q(x)})$$

In other words, it is the expectation of the logarithmic difference between the probabilities P and Q, where the expectation of P is already known.
- You'll need to perform a reparameterisation trick (since you can't calculate derivatives of samples.) in order to perform backpropagation. (you can't push gradients through a sampling node.)

$$z = \mu + \sigma \cdot \epsilon$$

where $\epsilon \sim \mathcal{N}(0, 1)$. You want to learn μ, σ.

1.6.1 Conditional Variational Autoencoders

Although VAEs are more robust when compared to their original autoencoder counterparts, the decoder class cannot produce outputs of a particular class on demand. CVAEs are an improved model of the original VAE architecture by conditioning on another description of the data, a class descriptor y.

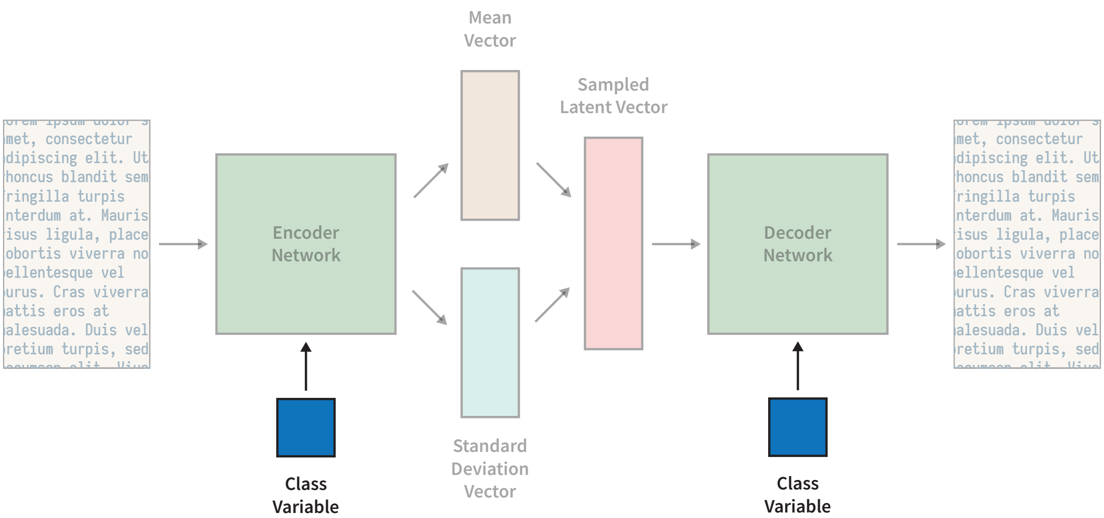

Figure 1.6: A model architecture for a CVAE, which includes the label being fed into the encoder and decoder networks.

During training time, a class (represented by some arbitary vector) is fed at the same time to the encoder and decoder. To generate an output that depends on y we feed that number to the decoder along with a random point in the latent space sampled from a standard normal distribution.
Samples can be generated from the conditional distribution $p(x|y)$. By changing the value of y, we can get corresponding samples $x \sim p(x|y)$. The system no longer relies on the latent space to encode what output is necessary; instead the latent

space encodes other information that can distinguish itself based on the differing y values.

1.7 Variational Autoregressive Decoders

Introduced by Du et al. (2018), Variational Autoregressive Decoders (VADs) attempt to circumvent the sampling problem introduced from CVAEs by introducing multiple latent variables into the autoregressive Decoder. At different time-steps, this allows the decoder to produce a multimodal distribution of text sequences, allowing a variety of responses to be produced.

VADs use the seq2seq architecture as the base with variable-length queries $x = \{x_1, x_2, ..., x_n\}$, and $y = \{y_1, y_2, ..., y_n\}$ representing the input and output responses respectively. The encoder network is a Bidirectional RNN with GRUs. The decoder network is an unidirectional RNN with GRUs. For each timestep t, each GRU in the decoder network is encoded with hidden state h_t^d.

Sequential Model

Sequential

Prior Model

Sequential Bag of Words

Learning Mechanism

VAD's propose

Bibliography

Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural Machine Translation by Jointly Learning to Align and Translate. *arXiv:1409.0473 [cs, stat]*. arXiv: 1409.0473. pages 4, 5

Bengio, Y., Ducharme, R., and Vincent, P. (2001). A Neural Probabilistic Language Model. pages 932–938. MIT Press. pages 2

Cho, K., van Merrienboer, B., Bahdanau, D., and Bengio, Y. (2014). On the Properties of Neural Machine Translation: Encoder-Decoder Approaches. *arXiv:1409.1259 [cs, stat]*. arXiv: 1409.1259. pages 4

Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. (2014). Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *arXiv:1412.3555 [cs]*. arXiv: 1412.3555. pages 4

Du, J., Li, W., He, Y., Xu, R., Bing, L., and Wang, X. (2018). Variational Autoregressive Decoder for Neural Response Generation. pages 3154–3163, Brussels, Belgium. Association for Computational Linguistics. pages 8

Dyer, C. (2017). Conditional Language Modelling. pages 1

Karpathy, A. (2015). The Unreasonable Effectiveness of Recurrent Neural Networks. pages 3

Le, J. (2018). Recurrent Neural Networks: The Powerhouse of Language Modeling. pages 2

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to Sequence Learning with Neural Networks. *arXiv:1409.3215 [cs]*. arXiv: 1409.3215. pages 4

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention Is All You Need. *arXiv:1706.03762 [cs]*. arXiv: 1706.03762. pages 5, 6