A Word Level Sequence to Sequence model for Dialogue Learning

Nikolaos Manginas

November 2019

Abstract

Historically the task of modelling natural language has been challenging to mathematically formulate. Early conversational agents were explicitly rule based with a extensive and precise human tailored inference engine, which allowed them to mimic human-level communication. The complexity of such systems however often resulted in sub-par performance. Modelling language is an cumbersome task and dialogue learning is one of its most challenging subfields, since input sequences of arbitrary length have to be processed and then accurate responses again of arbitrary length have to be constructed. Apart from the challenge of processing data of arbitrary size, accurate responses have dependencies far into the past which further increases the complexities of algorithms able to effectively model this convoluted feature space. Despite the above, cutting edge deep learning solutions have been successful in modelling this domain and have far surpassed any previous approaches. We demonstrate the implementation of the, as the last half decade, status-quo in dialogue learning, sequence to sequence models[1] which are often embellished with attention mechanisms[2]. We train out algorithms on the MetaLWoz dataset. Performance of the algorithms is up to par for subsets of the data but convergence is challenging on a dataset of this size in reasonable time without extremelly computationally capable hardware.

1 Introduction

The past decade has seen an extremely rapid growth in the development of deep learning based applications in both industry and academia. The justified mystery which initially surrounded the field, did not adversely effect the integration of such architectures into the forefront of the tech world. One of the successes of deep learning has been in Natural Language Processing within which dialogue learning resides. However, architectures used in this domain show significant variation over vanilla neural networks, the latter suffering drawbacks when applied to language based tasks. Instead, reccurent neural networks are often used which excel at modelling sequences by incorporating an internal state acting, intuitively, as memory.
Recurrent neural networks extend more basic architectures to model non static spaces. Language is such a task. The meaning or the generation of text is not based on some static input but is based on a sequence ranging back in time which dictates structure in the present. If we consider a basic neural network as a building block which takes some input and maps it to some output, recurrent networks further utilize a hidden state. This state incorporates any relevant past information which might be useful in mapping present inputs[3].

2 Sequence to Sequence Models

2.1 Overview

Neural networks are excellent function approximators. That is, given an set of inputs I and a set of corresponding outputs O, they are able to accurately approximate a function $f : I \to O$ which is able to map inputs previously not presented to the model, correctly. The methodology of effectively approximating this function will be discussed later but it is important to note here that for an accurate map, the model needs to be presented with an abundance of data accurately sampled from the distribution of the domain to which it is applied. In other words, if a model is to be constructed for classifying images of cats and dogs, it must be provided with data for both. This is near impossible for even the most basic conversational agents. Even if the agent is to converse only with one sentence at a time, the set of all possible sentences is infinite and the feature space quickly becomes to large in dimensions to model. We therefore model words as inputs, the set of which is enumerable. However a word cannot be expected to be sufficient for inference of a reply. We use recurrent networks to add state to the computation of our models and map input to output sequences [1]. These models are called sequence to sequence models and comprise of an encoder and a decoder.

2.2 Encoder

The encoder is a classic recurrent neural network. It is presented with an input sequence representing a sentence. Naturally the sequence is a list of integers corresponding to words in some vocabulary. The sentence is processed one word at a time and at each time step an output as well as new state is calculated by the reccurent network core. The process is repeated for each element in the input sequence and the final state is the encoding of the input sequence and is used for generation of a reply by the decoder. Thoughout these implementations we use GRU's for our recurrent architecture to relieve some computation weight from the hardware, instead of using more traditional LSTM's.

2.3 Decoder

The architecture of the decoder can in principle be as trivial as that of the encoder but is often embellished. At its simplest form it also is a simple recurrent neural network. However, in contrast to the encoder the outputs are now of significance since they are used to construct the output sequence. The decoder at the initial time step is presented with the encoding, the final state of the encoder, and an input reserved to signal the beginning of a sentence. It is then propagated though at each time step yielding an output and the next hidden state. Here, there is some extra logic involved since the output needs to map to our vocabulary and the network therefore also includes a fully connected layer mapping to the size of the vocabulary. At each time step the softmax function is applied to the output of the linear layer to yield a probability distribution over the vocabulary. Intuitively, this is the likelihood of each word occurring at the present time step. The predicted word is then fed to the decoder as input in the next time step. This process is repeated until the decoder produces a token signalling the end of the sequence.
Often, and in our implementation the decoder also utilizes an attention mechanism to further aid in sustaining long term dependencies. A precise formulation is beyond the scope, but can be found in the repository (Section 4). Essentially all outputs of the encoder, which are equivalent to the hidden state calculated at each encoder time step are weighted with respect to the current decoder input and state using a linear layer and their summation is passed along with the current input to the decoder core.

2.4 Training

Training is done on the decoder by using a variant of standard multi class cross entropy called negative likelihood. At each decoding time step the loss is calculated and the summation of the individual losses comprises the final loss for the input output sequence. Optimization is done with stochastic gradient descent. However, the training process is extremely computationally expensive since the gradients span back to the encoder for each output step.
Parallelization is crucial as backpropagating though the whole sequence in parallel exponentially increases convergence time. However, in order to process sequences in parallel, all samples at a given iteration must have exactly the same dimensions. That means we must check all possible dimensions of a pair and parallelize the ones with exactly equal dimensions. This approach greatly reduces training time.

3 Conclusion

We have used the MetaLWoz dataset which includes pairs of questions and answers in a variety of conversational topics to train an end-to-end conversational agent. By utilizing the power of sequence to sequence models as well as attention we have constructed an agent able to efficiently learn from data. However, the dataset is extremely large with close to 100000 question and answer pairs even with strict constraints on sentence length. Training on the dataset as a whole is thus extremely computationally expensive. Training on subsets of the data, converges to a satisfactory level extremely quickly which indicates that given considerable training and better hardware the same algorithm should converge on larger data volumes.

4 Repository

Code for the project can and sample convertations can be found on: https://github.com/nickmagginas/Chatbot.

References

[1] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proc. NIPS*, (Montreal, CA), 2014.

[2] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *CoRR*, vol. abs/1706.03762, 2017.

[3] A. Sherstinsky, "Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network," *CoRR*, vol. abs/1808.03314, 2018.