Sequence stacking using dual encoder Seq2Seq recurrent networks

Alessandro Bay
Cortexica Vision Systems Ltd.
London, UK

Biswa Sengupta*
Imperial College London
London, UK

Abstract

A widely studied non-polynomial (NP) hard problem lies in finding a route between the two nodes of a graph. Often meta-heuristics algorithms such as A^* are employed on graphs with a large number of nodes. Here, we propose a deep recurrent neural network architecture based on the Sequence-2-Sequence model, widely used, for instance in text translation. Particularly, we illustrate that utilising a context vector that has been learned from two different recurrent networks enables increased accuracies in learning the shortest route of a graph. Additionally, we show that one can boost the performance of the Seq2Seq network by smoothing the loss function using a homotopy continuation of the decoder's loss function.

1 Introduction

In the intersection of discrete optimization and graph theory lies an age-old problem of finding shortest routes between two nodes of a graph. Many theoretical properties of such shortest path algorithms can be understood by posing them on a graph [Sedgewick and Wayne, 2011]. Such graphs can be an inventory delivery algorithm posed on a road network graph (transportation) to a clustering of similar images and videos (computer vision). Traditionally, such discrete non-polynomial hard optimisation problems are studied using meta-heuristics algorithms such as the A^* algorithm. Other algorithms of notable mention are the Dantzig-Fulkerson-Johnson algorithm [Dantzig et al., 1954], branch-and-cut algorithms [Naddef and Rinaldi, 2001], neural networks [Ali and Kamoun, 1993], etc. Recent work [Bay and Sengupta, 2017] have proposed that recurrent neural networks can also be utilised in approximating the shortest routes produced by an A^* algorithm.

The primary problem surrounding the recurrent neural network's approximation of the shortest route problem is the difficulty of the network to encode longer sequences. This problem has been partly alleviated with network architectures such as long short-term memory (LSTM, Hochreiter and Schmidhuber [1997]) and the gated recurrent units (GRU, Cho et al. [2014]). Efforts have also been put towards a Neural Turing Machines [Graves et al., 2014] and a differentiable neural computer [Graves et al., 2016] that act as an augmented RNN with a (differentiable) external memory which can selectively be read or written to.

In this paper, we formulate a novel recurrent network based on the Sequence-to-Sequence (Seq2Seq, Sutskever et al. [2014]) architecture for increasing the fidelity of meta-heuristic approximations. Particularly, we show that using context vectors that have been generated by two different recurrent networks can facilitate the decoder to have an increased accuracy in approximating the shortest route estimated by the A^* algorithm.

*b.sengupta@imperial.ac.uk

2 Methods

In this section, we describe the data-sets, the procedure for generating the routes for training/test datasets, and the architecture of the dual encoder Seq2Seq network that forms the novel contribution of this paper:

2.1 Datasets

The graph is based on the road network of Minnesota[2]. Each node represents the intersections of roads while the edges represent the road that connects the two points of intersection. Specifically, the graph we considered has 376 nodes and 455 edges, as we constrained the coordinates of the nodes to be in the range $[-97, -94]$ for the longitude and $[46, 49]$ for the latitude, instead of the full extent of the graph, i.e., a longitude of $[-97, -89]$ and a latitude of $[43, 49]$, with a total number of 2,642 nodes.

2.2 Algorithms

The A^* meta-heuristics

The A^* algorithm is a best-first search algorithm wherein it searches amongst all of the possible paths that yield the smallest cost. This cost function is made up of two parts – particularly, each iteration of the algorithm consists of first evaluating the distance travelled or time expended from the start node to the current node. The second part of the cost function is a heuristic that estimates the cost of the cheapest path from the current node to the goal. Without the heuristic part, this algorithm operationalises the Dijkstra's algorithm [Dijkstra, 1959]. There are many variants of A^*; in our experiments, we use the vanilla A^* with a heuristic based on the Euclidean distance. Other variants such as Anytime Repairing A^* has been shown to give superior performance [Likhachev et al., 2004].

Paths between two randomly selected nodes are calculated using the A^* algorithm. On an average, the paths are 19 hops long and follow the distribution represented by the histogram in Figure 1.

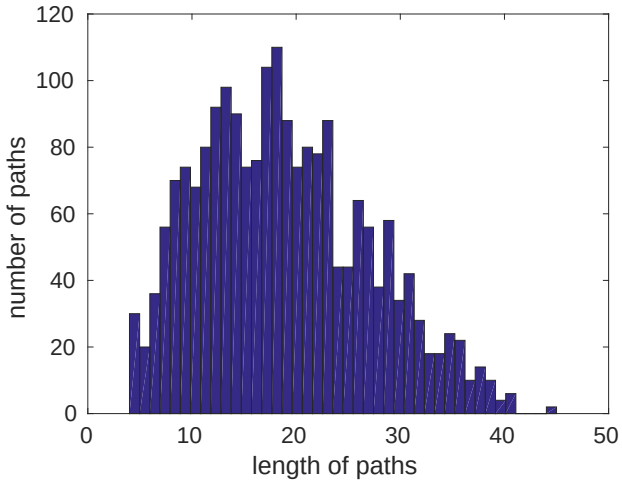


Figure 1: **Distribution of path lengths.** After selecting two nodes uniformly at random, we compute the shortest paths using the A^* algorithm. The average path length is 19 hops.

Recurrent deep networks

We utilised a variety of Sequence-to-Sequence recurrent neural networks for shortest route path predictions:

[2] https://www.cs.purdue.edu/homes/dgleich/packages/matlab_bgl

- An LSTM2RNN, where the encoder is modelled by an LSTM, i.e.

$$
\begin{aligned}
i(t) &= \text{logistic}\Big(A_i x(t) + B_i h(t-1) + b_i\Big) \\
j(t) &= \tanh\Big(A_j x(t) + B_j h(t-1) + b_j\Big) \\
f(t) &= \text{logistic}\Big(A_f x(t) + B_f h(t-1) + b_f\Big) \\
o(t) &= \text{logistic}\Big(A_o x(t) + B_o h(t-1) + b_o\Big) \\
c(t) &= f(t) \odot c(t-1) + i(t) \odot j(t) \\
h(t) &= o(t) \odot \tanh\Big(c(t)\Big),
\end{aligned}
$$

  while the decoder is a vanilla RNN, i.e.

$$
\begin{cases}
h(t) = \tanh(Ax(t) + Bh(t-1) + b) \\
y(t) = \text{logsoftmax}(Ch(t) + c)
\end{cases}
. \tag{1}
$$

- A GRU2RNN, where the encoder is modelled by a GRU, i.e.

$$
\begin{aligned}
z(t) &= \text{logistic}\Big(A_z x(t) + B_z h(t-1) + b_z\Big) \\
r(t) &= \text{logistic}\Big(A_r x(t) + B_r h(t-1) + b_r\Big) \\
\tilde{h}(t) &= \tanh\Big(A_h x(t) + B_h(r(t) \odot h(t-1)) + b_h\Big) \\
h(t) &= z(t) \odot h(t-1) + (1 - z(t)) \odot \tilde{h}(t),
\end{aligned}
$$

  while the decoder is again a vanilla RNN, as in Equation (1).

- A dual context Seq2Seq model, where two different latent representations are learnt using two different encoders (one LSTM and one GRU). The context vector takes the form of a stacked latent encoding. In Figure 2, we show the two context vectors stacked in a matrix for each path in the training set. For both encoders, their respective matrices are full rank and also the stacked one is of full rank. This means that GRU and LSTM encode very different context vectors and it is worth considering them both for an accurate encoding.

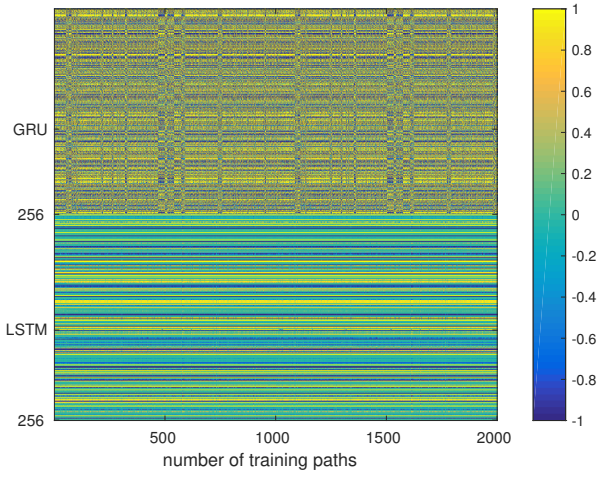
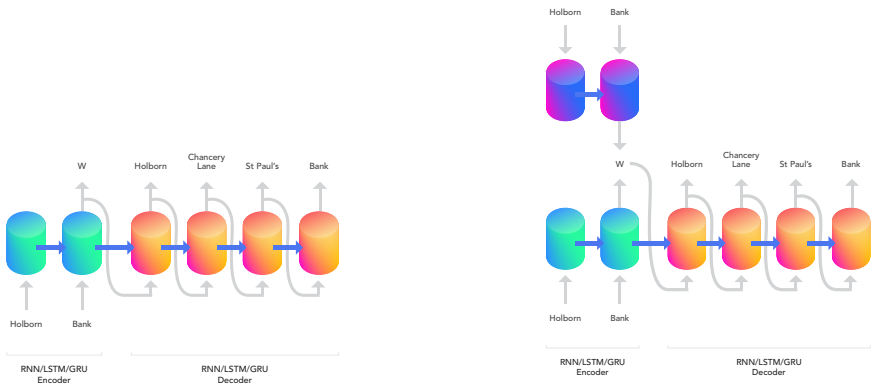


Figure 2: **Context vectors for GRU and LSTM encoders.** Matrices with training context vectors for GRU and LSTM. Their individual and composite rank are full.

- A dual context Seq2Seq model, where two different latent representations are learnt using two different encoders (one LSTM and one GRU) and the decoder is represented by a vanilla RNN, trained with homotopy continuation [Vese, 1999]. This is done by convolving the loss function with a Gaussian kernel – for more details please refer to Bay and Sengupta [2017]. Our novel contribution lies in extending the framework of Mobahi [2016] by obtaining an analytic approximation of the log softmax function. Table 1 illustrates the diffused forms of the most popular activation functions.

function	original	diffused
error	$\text{erf}(\alpha x)$	$\text{erf}\left(\frac{\alpha x}{\sqrt{1+2(\alpha\sigma)^2}}\right)$
tanh	$\tanh(x)$	$\tanh\left(\frac{x}{\sqrt{1+\frac{\pi}{2}\sigma^2}}\right)$
sign	$\begin{cases} +1 & \text{if } x > 0 \\ 0 & \text{if } x = 0 \\ -1 & \text{if } x < 0 \end{cases}$	$\text{erf}\left(\frac{x}{\sqrt{2}\sigma}\right)$
relu	$\max(x, 0)$	$\frac{\sigma}{\sqrt{2\pi}} \exp\left(\frac{-x^2}{2\sigma^2}\right) + \frac{1}{2}x\left(1 + \text{erf}\left(\frac{x}{\sqrt{2}\sigma}\right)\right)$
logsoftmax	$x - \log\left(\sum \exp(x)\right)$	$\left(\left(1 - \frac{1}{\pi}\right)\exp\left(-\pi\sigma^2\right) + \frac{1}{\pi}\right)x - \log(\sum(\exp(x)))$

Table 1: **List of diffused forms (Weierstrass transform).** We report the most popular non-linear activation functions along with their diffused form. This is obtained by convolving the function with the heat kernel $K(x, \sigma)$. This table extends the work in Mobahi [2016] by an analytic approximation of the log softmax function. For more details please refer to Bay and Sengupta [2017].



(a) Seq2Seq network

(b) Dual-context Seq2Seq network

Figure 3: **Dual-context Sequence-to-Sequence architecture for approximating the A^* meta-heuristics.** For both networks, the first two modules on the left are the encoder while the last four represent the decoded output, representing the shortest route between Holborn and Bank. The network is trained using shortest route snippets that have been generated using an A^* algorithm. w represents the context vector.

For all networks, as shown in Figure 3, the input is represented by the [source, destination] tuple, which is encoded in a context vector (w) and subsequently decoded into the final sequence to obtain the shortest path connecting the source to the destination. Moreover, during the test phase, we compute two paths, one from the source to the destination node and the other from the destination to the source node, that forms an intersection to result in the shortest path.

3 Results

For the graph of Minnesota with 376 nodes and 455 edges, we generated 3,000 shortest routes between two randomly picked nodes using the A^* algorithm. We used these routes as the training set for the Seq2Seq algorithms using a 67-33% training-test splits.

For the two encoders involved, we choose a hidden state with 256 units, such that the joint latent dimension of the two neural networks is 512. In our experiments, we compare the standard Seq2Seq with either 256 or 512 hidden units. We run the training for 400 epochs, updating the parameters with an Adam optimisation scheme [Kingma and Ba, 2014], with parameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$, starting from a learning rate equal to 10^{-3}. On the other hand, for the diffused loss function, we smooth the cost function using a Gaussian kernel of standard deviation $s = \{30, 5, 1, 0.0001\}$. The training iterates converged after 100 epochs for each value of s.

The prediction accuracy on the test data-set is reported in Table 2. As we can see, doubling the hidden state dimension marginally increases the percentage of shortest paths (1%) and the successful paths, that are not necessarily the shortest (0.2% and 1.6% for GRU and LSTM encoders, respectively). Alternatively, our proposed dual encoder achieves improvement on the shortest paths (almost 58%). If trained with diffusion (homotopy continuation), it turns out to be the best performing algorithm with about 60% of accuracy on the shortest paths and more than 78% on the successful cases.

method	shortest	successful
LSTM2RNN (256)	47%	69.5%
LSTM2RNN (512)	48%	71.1%
GRU2RNN (256)	48%	73.1%
GRU2RNN (512)	49%	73.3%
dual encoder	57.7%	77.1%
dual encoder with diffusion	**59.6%**	**78.3%**

Table 2: **Results on the Minnesota graph.** Percentage of shortest path and successful paths (that are not necessarily shortest) are shown for a wide-variety of Seq2Seq models, with context vector dimension equal to either 256 or 512. All scores are relative to an A^* algorithm, that achieves a shortest path score of 100%.

4 Discussion

It is clear that using two context vectors instead of one improves the decoder's accuracy in approximating the A^* algorithm. What we have proposed in this paper is akin to feature stacking wherein two different sets of features are stacked to increase classification accuracy. Our experiments that control the embedding dimension of the latent context vector (256 or 512) show that the increased number of successful routes produced by the neural network is due to the encoding dynamics, not the encoding dimension. Indeed, a homotopy continuation induced diffusion increases the accuracy by $\approx 2\%$, it still falls short in improving the temporal memory of the encoder.

In future, we foresee using a sequential probabilistic model of the latent context vector that might afford to learn the structure of the sub-route's temporal congruency.

Acknowledgments

BS is thankful to the Issac Newton Institute for Mathematical Sciences for hosting him during the "Periodic, Almost-periodic, and Random Operators" workshop.

References

M. K. M. Ali and F. Kamoun. Neural networks for shortest path computation and routing in computer networks. *IEEE Transactions on Neural Networks*, 4(6):941–954, Nov 1993. ISSN 1045-9227.

A. Bay and B. Sengupta. Approximating meta-heuristics with homotopic recurrent neural networks. *ArXiv e-prints: 1709.02194*, September 2017.

Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

G Dantzig, R Fulkerson, and S Johnson. Solution of a large-scale traveling-salesman problem. *Operations Research*, 2:393–410, 1954.

Edsger W Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014.

Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwińska, Sergio Gómez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John Agapiou, et al. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538(7626):471–476, 2016.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8), November 1997.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Maxim Likhachev, Geoffrey J Gordon, and Sebastian Thrun. ARA*: Anytime A* with provable bounds on sub-optimality. In *Advances in Neural Information Processing Systems*, pages 767–774, 2004.

Hossein Mobahi. Training recurrent neural networks by diffusion. *arXiv preprint arXiv:1601.04114*, 2016.

D. Naddef and G. Rinaldi. The vehicle routing problem. chapter Branch-and-cut Algorithms for the Capacitated VRP, pages 53–84. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2001. ISBN 0-89871-498-2.

Robert Sedgewick and Kevin Wayne. *Algorithms*. Addison-Wesley Professional, 4th edition, 2011.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

Luminita Vese. A method to convexify functions via curve evolution. *Communications in partial differential equations*, 24(9-10):1573–1591, 1999.