Pointer Networks

Roman Ring

Institute of Computer Science
University of Tartu
roman.ring@ut.ee

December 18, 2018

Abstract

Recently progress was made to extend Recurrent Neural Networks (RNNs) to support efficiently mapping arbitrary sequences [1], and even developing fully differentiable turing machines [2]. However, one major drawback of these approaches is the fixed length of the sequences. Authors of Pointer Networks [3] propose a solution to this issue, and show results on three classical algorithmic tasks: convex hull, Delaunay triangulation, and travelling salesman problem (TSP). Authors' promising results imply possibility of eventually applying RNNs to more fundamental problems in computer science such as sorting and searching.

I. Introduction

Recurrent Neural Network (RNN) [4] is a class of non-linear function approximators, specialized for data with a temporal component, e.g. sentences of characters, and are particularly popular in the domain of natural language processing (NLP). Recently, two major breakthroughs have occurred in this domain: the sequence-to-sequence model that enabled learning efficient mappings from one sequence to another [1], and the so-called 'attention' mechanism, which significantly improved quality of generated output by learning relative importance of various input sequence parts, measured by their effect on the resulting error in the final prediction [5].

Since their inception, the idea of applying RNNs to algorithmic problems has puzzled researchers from a wide range of backgrounds with varying success of their proposed solutions [6]. However, even the more successful approaches were limited in scope as they relied on the fixed length of input and expected output sequences. The goal of this essay is to review a relatively simple, yet efficient solution to the problem that relies on reusing the attention mechanism [3].

II. Background

Mathematically, sequence to sequence model can be viewed as an estimate of the conditional probability $p(Y|X;\theta)$, where Y is the target output sequence, X is the input sequence, and θ are the RNN parameters. The probability is estimated via the chain rule:

$$p(Y|X;\theta) = \prod_{t=1}^{T} p(y_t|h_\theta(X), y_1, \ldots, y_{t-1}; \theta)$$

where input sequence X is compressed into an internal representation through $h_\theta(X)$. Parameters θ are optimized with gradient descent given some pre-defined training set. Note that estimated probability is over the entire fixed-length output sequence $Y = (y_1, \ldots, y_T)$.

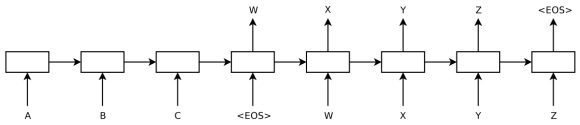

Figure 1: model encodes input X to internal $h_\theta(X)$, uses it and all previous outputs to generate next [1].

Attention vector a is generated from u, a combination of $h_\theta(X)$, encoder and decoder hidden states (represented by boxes in Figure 1), which is then passed through the softmax function $a_i = \frac{e^{u_i}}{\sum e^{u_j}}$. It is then applied to encoder hidden states as dot product $a \cdot e$. Intuitively, attention mechanism helps RNN by focusing on relevant subsets of the input through importance weights.

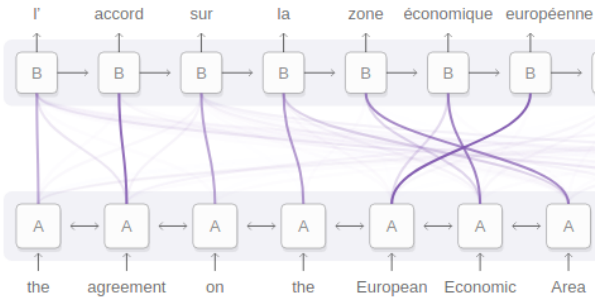

Figure 2: Output words rely on full input sentence with attention represented by intensity of the line color [7].

III. Methods

Even when combined with the attention mechanism, RNNs are not versatile enough when applied to algorithmic problems such as TSP due to the fixed length nature of the architecture.

To remedy this issue, authors of Pointer Networks make use of the attention mechanism in a different way: rather than act as input importance weights for the decoder, the attention mechanism is used directly as a probability distribution over the input elements.

$$p(y_t|h_\theta(x_1,\ldots,x_T),y_1,\ldots,y_{t-1}) = a_t$$

From this distribution a pointer (index) to the input element is obtained by taking the *argmax*.

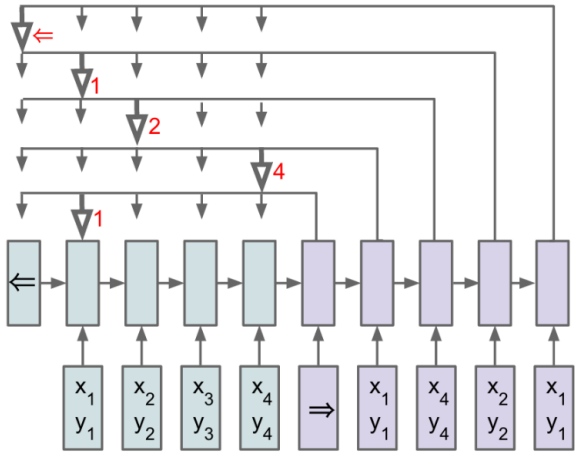

Figure 3: Pointer Networks: output sequence is generated as a series of pointers to the input sequence elements [3].

IV. Results

Authors evaluate proposed method on three fundamental combinatorics problems: Delaunay Triangulation (a subset of points on a plane such that each triangles circumcircle is empty), Convex Hull (a subset of points such that their perimeter covers the whole set), and Traveling Salesman Problem (a unique tour on a weighted graph such that travel cost is minimized).

Authors implementation completely solves the Convex Hull problem, although same can be said about the baseline. What's perhaps more impressive is that authors have shown their main hypothesis holds on this problem in the sense that implemented network was able to handle variable length inputs as well, unlike baseline solutions. Moreover, Pointer Networks were capable of extrapolating to larger length inputs than seen at training time.

On Delaunay Triangulation Pointer Networks have significantly outperformed the baseline solutions. Measured by accuracy of predicted triangulation, Pointer Networks were up to 35% more accurate than baselines.

The most impressive result was perhaps on the Traveling Salesman Problem where Pointer Networks were not only capable of solving the problem, but in fact they were able to keep up with classical algorithmic search based implementations.

Problem	Baseline	Ptr-Net
Convex Hull	99.7%	99.9%
Delaunay Tr.	38.9%	72.6%
TSP	3.85	3.88

Table 1: Pointer Network results on three combinatorial problems. Convex Hull and Delaunay Triangulation measured by accuracy of prediction, TSP measured by tour length. Baselines for Convex Hull and Delaunay Triangulation are seq-to-seq models with attention, whereas on TSP it is classical search based algorithm.

V. Conclusion

Authors of the Pointer Networks have proposed a relatively simple extension to state-of-the-art approaches in NLP and have shown that their solution can perform well on classical combinatorial problems.

A promising next step would be exploring even more fundamental and difficult problems, eventually scaling up to general differentiable finite state machines.

References

[1] 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.

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

[3] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In *Advances in Neural Information Processing Systems*, pages 2692–2700, 2015.

[4] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[5] Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. End-to-end memory networks. In *Advances in neural information processing systems*, pages 2440–2448, 2015.

[6] Anthony J Robinson. An application of recurrent nets to phone probability estimation. *IEEE transactions on Neural Networks*, 5(2):298–305, 1994.

[7] Chris Olah and Shan Carter. Attention and augmented recurrent neural networks. *Distill*, 2016.