Pointer Networks

NeurIPS 2015

Oriol Vinyals*, Meire Fortunato*, Navdeep Jaitly†

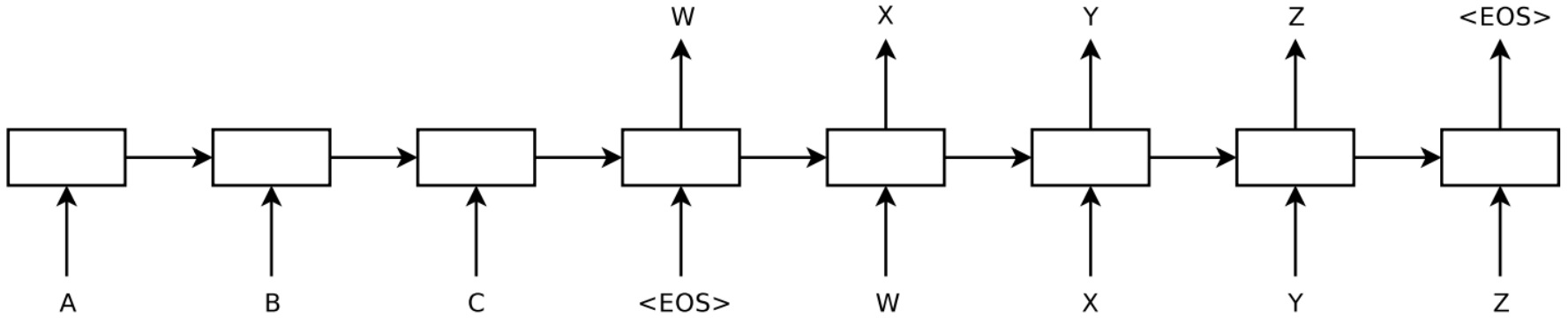
***Google Brain**

†Department of Mathematics, UC Berkeley

Part 1.

Introduction

- **Recurrent Neural Networks (RNNs)**
  - This architecture limited them to settings where the inputs and outputs were available at a fixed frame rate
- **Sequence-to-sequence paradigm**
  - Remove these constraints by using one RNN to map an input sequence to an embedding and another (possibly the same) RNN to map the embedding to an output sequence

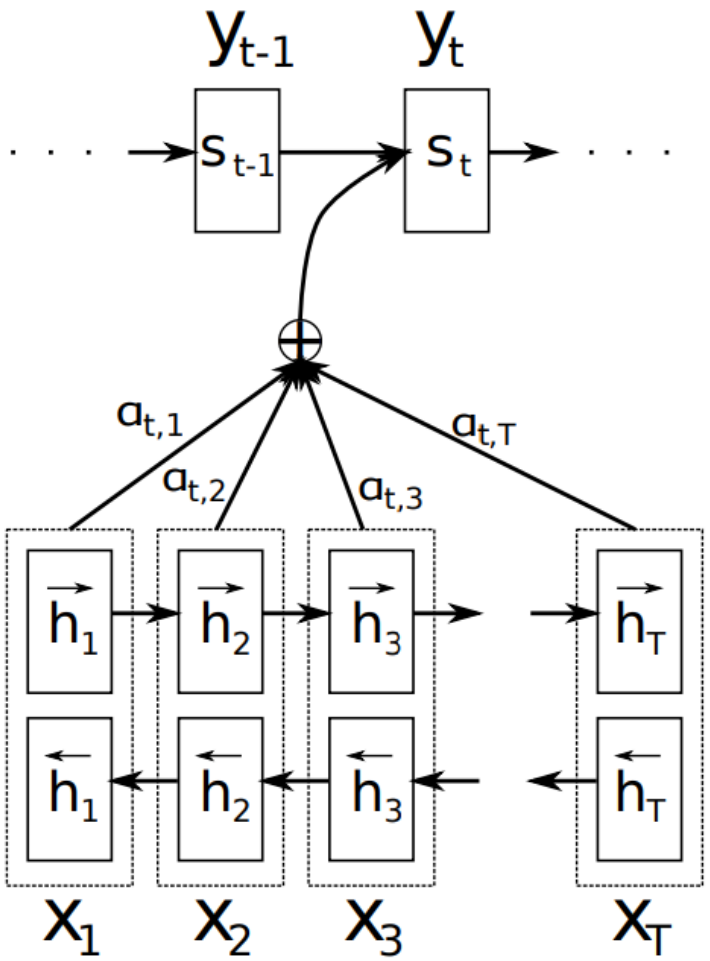


Introduce many short term dependencies in the data that make the optimization problem much easier

Ilya Sutskever et al. Sequence to sequence learning with neural networks. NIPS. 2014.

Introduction

- **Bahdanau et. al., 2015**



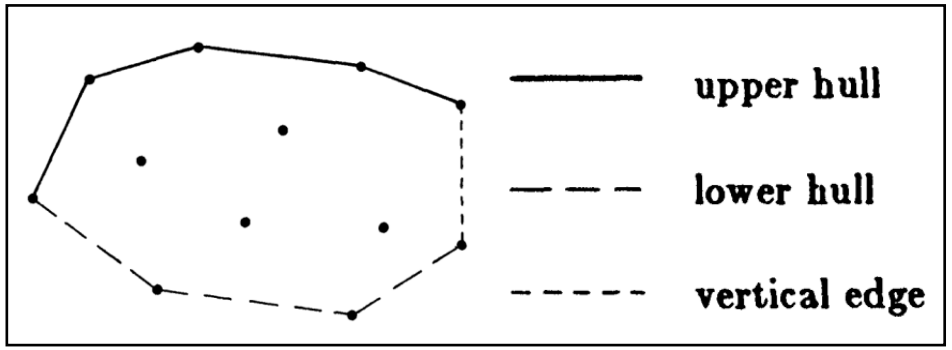
- Augment the decoder by propagating extra contextual information from the input using a content-based attentional mechanism
  - Make it possible to apply RNNs to new domains, achieving SOTA results in NLP
- Still require the size of the output dictionary to be fixed a priori
  - Cannot directly apply this framework to combinatorial problems where the size of the output dictionary depends on the length of the input sequence

Dzmitry Bahdanau et al. Neural machine translation by jointly learning to align and translate. ICLR. 2015.

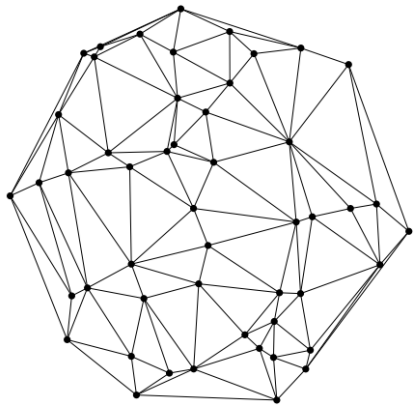
Part 1.

Introduction

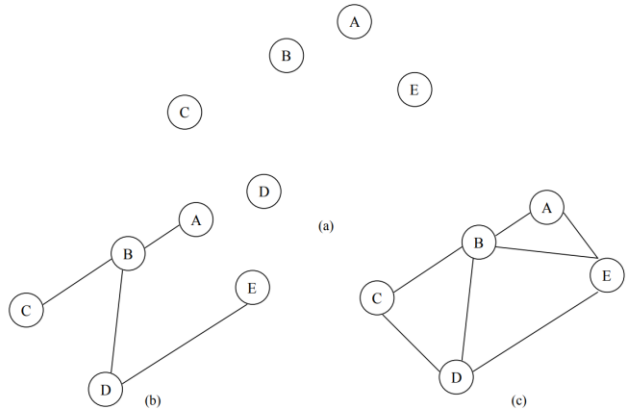
- **Address this limitation**
  - Repurpose the attention mechanism of [5] to create pointers to input elements
  - Proposed Method: Pointer Networks (Ptr-Nets) with combinatorial optimization problems
    - Computing planar convex hulls
    - Delaunay triangulations
    - The symmetric planar Travelling Salesman Problem (TSP)
  - Models produce approximate solutions to these problems in a purely data drive fashion



Convex Hull



Delaunay triangulation



The Traveling Salesman Problem

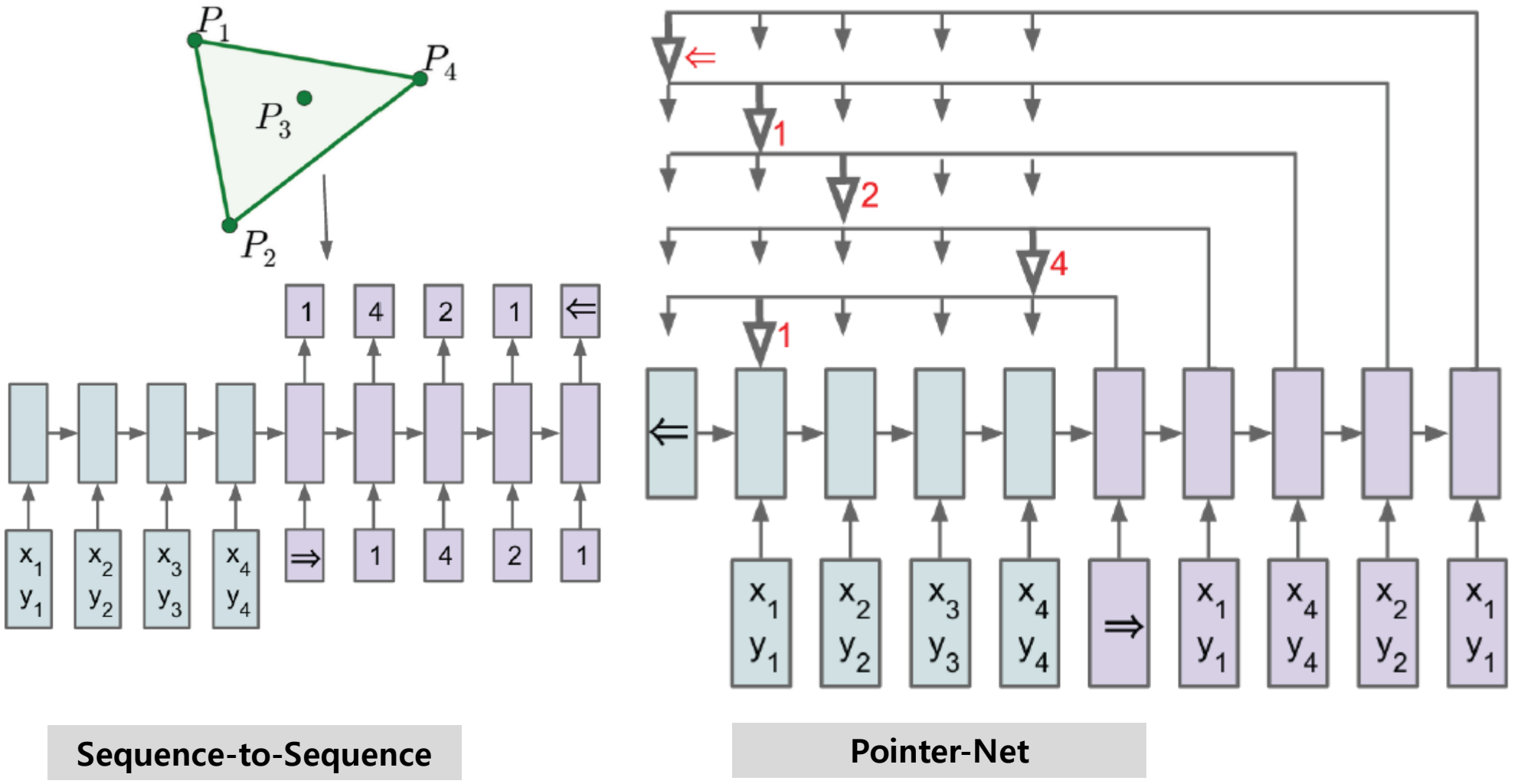
David G. Kirkpatrick et al. The Ultimate Planar Convex Hull Algorithm? SIAM J. COMPUT. 1986
Prof. Emo Welzl and Prof. Bernd Gärtner. Theory of Combinatorial Algorithms Chapter 6 Delaunay Triangulations. Swiss Federal Institue of Technology Zurich. 2013.
Prof. Ioannis Tollis. HY-583 Graph Algorithms Chapter 10 The Traveling Salesman Problem. University of Crete. 2021.

Introduction

- **Contribution**
  - Pointer Net is simple and effective
    - Deal with the fundamental problem of representing variable length dictionaries by using a softmax probability distribution as a "pointer"
  - Distinct non-trivial algorithmic problems involving geometry
    - The learned model generalizes to test problems with more points than the training problems
  - Learns a competitive small scale $(n \leq 50)$ TSP approximate solver
    - A purely data driven approach can learn approximate solutions to problems that are computationally intractable

Part 1.

Introduction



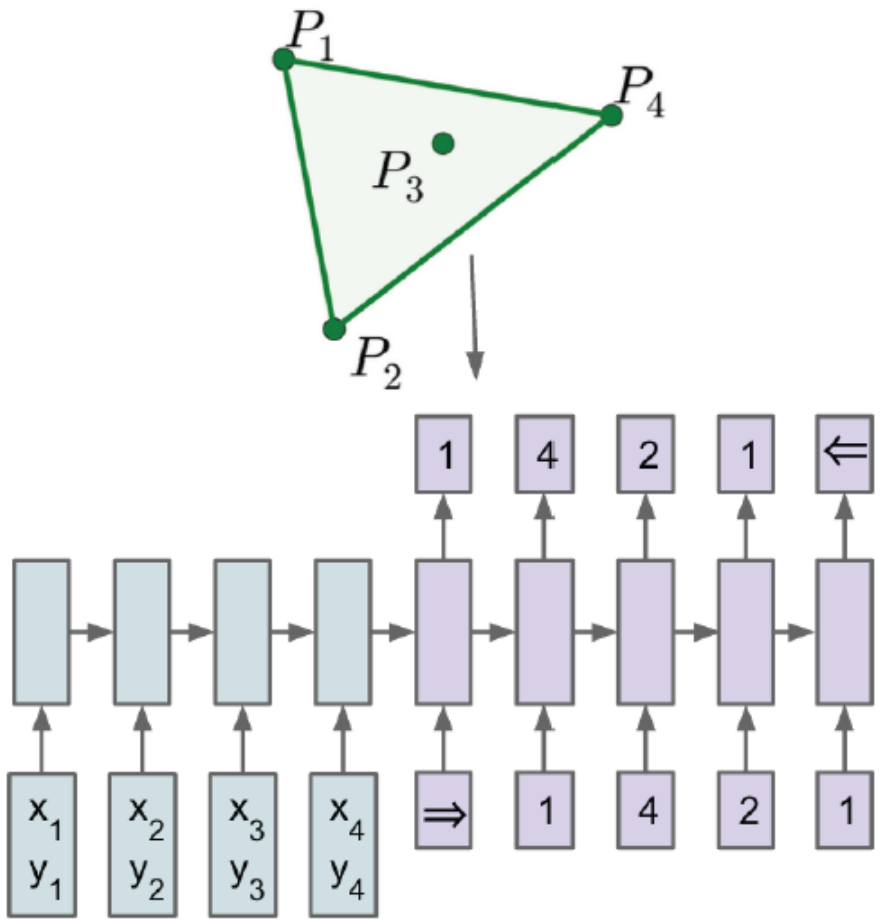
Sequence-to-Sequence

Pointer-Net

Peter Brown et al. The mathematics of statistical machine translation: Parameter estimation. Computational Linguistics, 19(2):263–311, 1993.

Sequence-to-Sequence Model

- **The conditional probability**



$$p(\mathcal{C}^{\mathcal{P}}|\mathcal{P};\theta) = \prod_{i=1}^{m(\mathcal{P})} p_\theta(C_i|C_1,\ldots,C_{i-1},\mathcal{P};\theta)$$

- A training pair $(\mathcal{P},\mathcal{C}^{\mathcal{P}})$
- Use a parametric model (an RNN with parameters θ)
  - Estimate the terms of the probability chain rule
- A sequence of n vectors

$$\mathcal{P} = \{P_1,\ldots,P_n\}$$

- A sequence of $m(\mathcal{P})$ indices, each between 1 and n

$$\mathcal{C}^{\mathcal{P}} = \{C_1,\ldots,C_{m(\mathcal{P})}\}$$

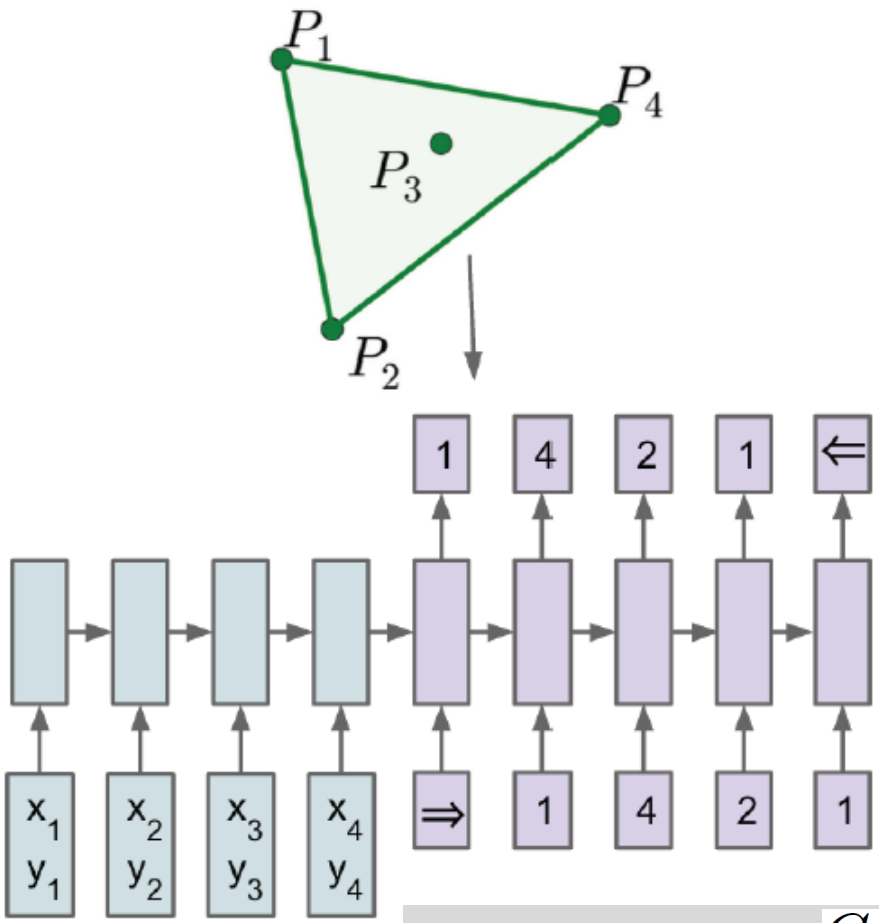
- Training Stage: Maximize the conditional probabilities

$$\theta^* = \arg\max_\theta \sum_{\mathcal{P},\mathcal{C}^{\mathcal{P}}} \log p(\mathcal{C}^{\mathcal{P}}|\mathcal{P};\theta)$$

Sequence-to-Sequence Model

- **The conditional probability**

$$p(\mathcal{C}^{\mathcal{P}}|\mathcal{P};\theta) = \prod_{i=1}^{m(\mathcal{P})} p_\theta(C_i|C_1,\ldots,C_{i-1},\mathcal{P};\theta)$$



◦ Use an Long Short Term Memory (LSTM)

$$p_\theta(C_i|C_1,\ldots,C_{i-1},\mathcal{P};\theta)$$

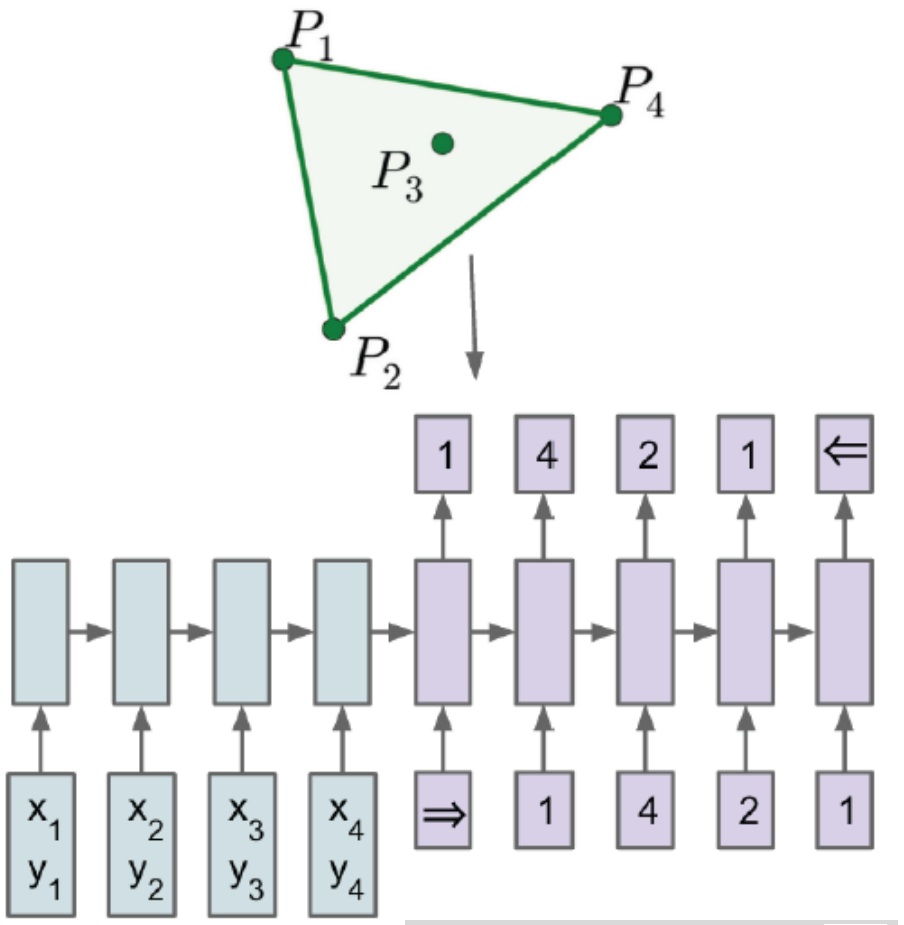
◦ The RNN is fed P_i at each time step, i , until the end of the input sequence is reached, at which time a special symbol, $\Rightarrow$ (input)

◦ Switches to the generation mode until the network encounters the special symbol, $\Leftarrow$ (termination)

No statistical independence assumptions
(The former, The latter RNN)

Sequence-to-Sequence Model

- **Sequence-to-sequence model**



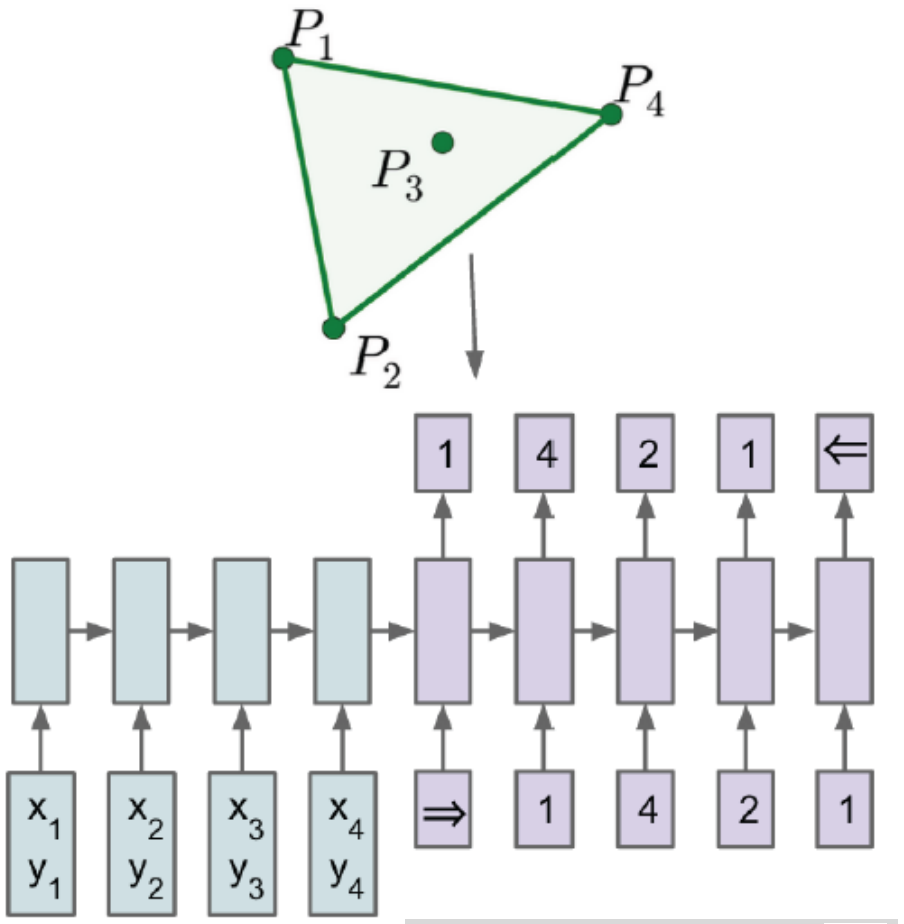
Training $$\theta^* = \arg\max_{\theta} \sum_{\mathcal{P},\mathcal{C}^{\mathcal{P}}} \log p(\mathcal{C}^{\mathcal{P}}|\mathcal{P};\theta)$$

Inference $$\hat{\mathcal{C}}^{\mathcal{P}} = \arg\max_{\mathcal{C}^{\mathcal{P}}} p(\mathcal{C}^{\mathcal{P}}|\mathcal{P};\theta^*)$$

- A sequence $\mathcal{P}$
- The learnt parameters θ^*
  - Select the sequence $\hat{\mathcal{C}}^{\mathcal{P}}$ with the highest probability
- Find the optimal sequence $\hat{\mathcal{C}}$
  - Problem - Computationally impractical:
    Combinatorial number of possible output sequences
  - Solution: Use a beam search procedure to find the best possible sequence given a beam size

Sequence-to-Sequence Model

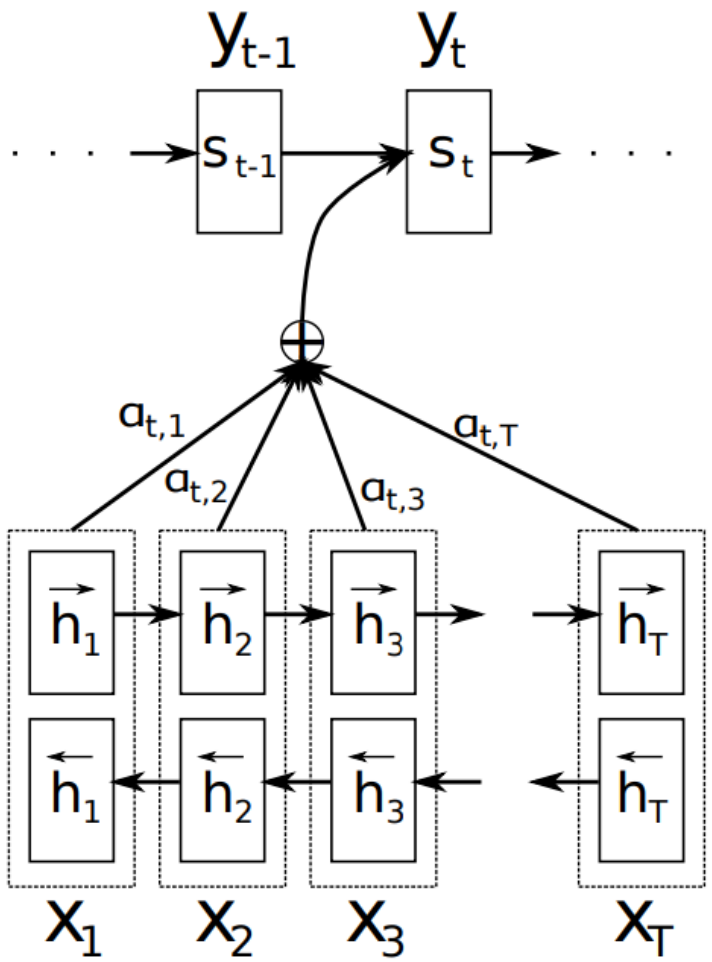
- **Training & Inference**



- The output dictionary size for all symbols C_i
  - It is fixed and equal to n, since the outputs are chosen from the input
- Need to train a separate model for each n
  - Prevents us from learning solutions to problems that have an output dictionary with a size that depends on the input sequence length
- The number of outputs $O(n)$
- Computational complexity $O(n)$
- Exact algorithms for the problems are more costly
  - $O(n \log n)$ For the convex hull problem

Part 3.

Content Based Input Attention



- **Vanilla sequence-to-sequence model**
  - Produce the entire output sequence $\mathcal{C}^{\mathcal{P}}$
    - Use the fixed dimensional state of the recognition RNN at the end of the input sequence $\mathcal{P}$
  - Constraint
    - Limit the amount of information and computation that can flow through to the generative model

- **Attention model of (Bahdanau et. al., 2015)**
  - Ameliorate this problem
    - Augment the encoder and decoder RNNs with an additional NN that uses an attention mechanism over the entire sequence of encoder RNN states

Dzmitry Bahdanau et al. Neural machine translation by jointly learning to align and translate. ICLR. 2015.

Content Based Input Attention

- **Compute the attention vector at each output time i**

$$
\begin{aligned}
u_j^i &= v^T \tanh(W_1 e_j + W_2 d_i) && j \in (1, \ldots, n) \\
a_j^i &= \text{softmax}(u_j^i) && j \in (1, \ldots, n) \\
d_i' &= \sum_{j=1}^{n} a_j^i e_j
\end{aligned}
$$

  - LSTM RNN: Use the state after the output gate has been component-wise multiplied by the cell activations
  - Softmax normalizes the vector u^i to be the "attention" mask over the inputs
  - Learnable parameters of the model: **A Vector** v **/ Square matrices** $W_1 W_2$
  - d_i' d_i are concatenated and used as the hidden states
    - Hidden states from which we make predictions and which we feed to the next time step in the recurrent model

Content Based Input Attention

- **Compute the attention vector at each output time i**

$$
\begin{aligned}
u_j^i &= v^T \tanh(W_1 e_j + W_2 d_i) && j \in (1, \ldots, n) \\
a_j^i &= \text{softmax}(u_j^i) && j \in (1, \ldots, n) \\
d_i' &= \sum_{j=1}^{n} a_j^i e_j
\end{aligned}
$$

  - Computational complexity $O(n^2)$
  - Performs significantly better than the seq2-to-seq2 model on the convex hull problem
  - Not applicable to problems where the output dictionary size depends on the input
  - Nevertheless, a very simple extension (or rather reduction) of the model allows us to do this easily

Ptr-Net (Pointer-Net)

- **Modification of the attention model**

$$u_j^i = v^T \tanh(W_1 e_j + W_2 d_i) \qquad j \in (1, \ldots, n)$$

$$p(C_i | C_1, \ldots, C_{i-1}, \mathcal{P}) = \text{softmax}(u^i)$$

  - Solve combinatorial optimization problems
    - Output dictionary size depends on the number of elements in the input sequence
    - Use a softmax distribution over a fixed sized output dictionary to compute $p(C_i | C_1, \ldots, C_{i-1}, \mathcal{P})$
  - Don't blend the encoder state e_j to propagate extra information to the decoder
  - Instead, use u_j^i as pointers to the input elements
  - Targets problems whose outputs are discrete and correspond to positions in the input
  - At inference, Don't respect the constraint that the outputs map back to the inputs exactly
  - Without constraints, the predictions are bound to become blurry over longer sequence

Motivation and Datasets Structure

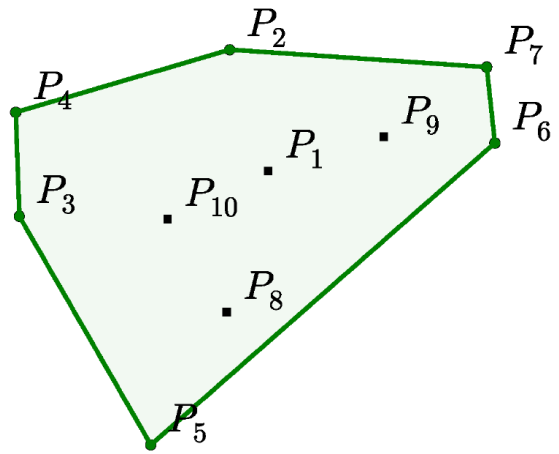
- **Review each of the three problems we considered**
  - The inputs are planar point sets
    - The cartesian coordinates of the points
  - Convex Hull:

    The Delaunay triangulation or the solution to the corresponding Travelling Salesman Problem
  - Sample from a uniform distribution in
  - Sequences representing the solution associated to the point set

Motivation and Datasets Structure

- **Input/output pair $(\mathcal{P}, \mathcal{C}^{\mathcal{P}})$ for task**

Convex Hull



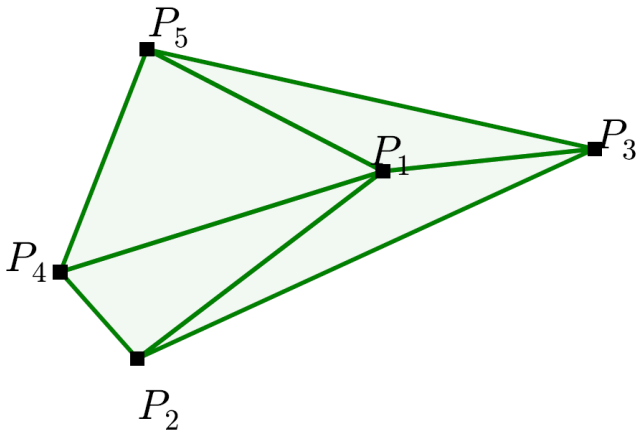
$$\mathcal{P} = \{P_1, \ldots, P_{10}\}$$

$$\mathcal{C}^{\mathcal{P}} = \{\Rightarrow, 2, 4, 3, 5, 6, 7, 2, \Leftarrow\}$$

Finding the solution has complexity

$$O(n \log n)$$

Delaunay Triangulation



$$\mathcal{P} = \{P_1, \ldots, P_5\}$$

$$\mathcal{C}^{\mathcal{P}} = \{\Rightarrow, (1,2,4), (1,4,5), (1,3,5), (1,2,3), \Leftarrow\}$$

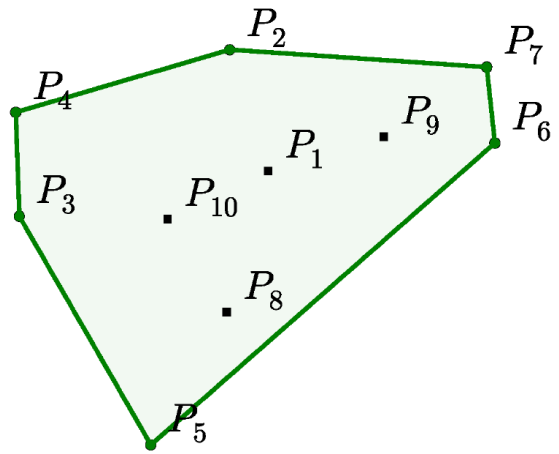
Finding the solution has complexity

$$O(n \log n)$$

Motivation and Datasets Structure

- **Input/output pair $(\mathcal{P}, \mathcal{C}^{\mathcal{P}})$ for task**

Convex Hull



$$\mathcal{P} = \{P_1, \ldots, P_{10}\}$$

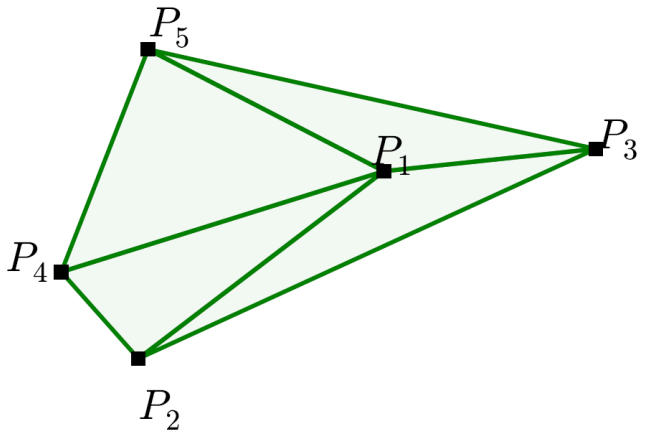
$$\mathcal{C}^{\mathcal{P}} = \{\Rightarrow, 2, 4, 3, 5, 6, 7, 2, \Leftarrow\}$$

- Vectors P_j are uniformly sampled from $[0, 1] \times [0, 1]$
- The elements C_i are indices between 1 and n corresponding to positions in the sequence $\mathcal{P}$
- Special tokens representing beginning or end of sequence
- To represent the output as a sequence
- Start from the point with the lowest index, and go counter-clockwise
  - This is an arbitrary choice but helps reducing ambiguities during training

Motivation and Datasets Structure

- **Input/output pair $(\mathcal{P}, \mathcal{C}^{\mathcal{P}})$ for task**

Delaunay Triangulation



$$\mathcal{P} = \{P_1, \ldots, P_5\}$$

$$\mathcal{C}^{\mathcal{P}} = \{\Rightarrow, (1,2,4), (1,4,5), (1,3,5), (1,2,3), \Leftarrow\}$$

- The outputs $\mathcal{C}^{\mathcal{P}} = \{C_1, \ldots, C_{m(\mathcal{P})}\}$
  The corresponding sequences representing the triangulation of the point set $\mathcal{P}$
- Each C_i is a triple of integers from 1 to n corresponding to the position of triangle vertices in $\mathcal{P}$ or the beginning/end of sequence tokens
- Without loss of generality, order the triangles C_i by their incenter coordinates (lexicographic order) and choose the increasing triangle representation
- Additionally each triangle representation C_i of three integers can also be permuted
- Without ordering, the models learned were not as good

Motivation and Datasets Structure

- **Travelling Salesman Problem (TSP)**
  - TSP arises in many areas of theoretical computer science and is an important algorithm used for microchip design or DNA sequencing
  - Focused on the planar symmetric TSP:
    given a list of cities, we wish to find the shortest possible route that visits each city exactl once and returns to the starting point
  - Assume the distance between two cities is the same in each opposite direction
  - NP-hard problem which allows us to test the capabilities and limitations of our model

Motivation and Datasets Structure

- **Travelling Salesman Problem (TSP):**
  Input/output pair $(\mathcal{P}, \mathcal{C}^{\mathcal{P}})$ for task
  - $\mathcal{P}$ will be the cartesian coordinates representing the cities, which are chosen randomly in the $[0,1] \times [0,1]$ square
  - $\mathcal{C}^{\mathcal{P}} = \{C_1, \ldots, C_{m(\mathcal{P})}\}$ will be a permutation of integers from 1 to n representing the optimal path (or tour)
  - For consistency, in the training dataset, we always start in the first city without loss of generality
  - To generate exact data, implement the Held-Karp algorithm
    - Finds the optimal solution in $O(2^n n^2)$
    - For larger n, producing exact solutions is extremely costly, therefore we also considered algorithms that produce approximated solutions

Experiment

- **Architecture and Hyperparameters**
  - Even though there are likely some gains to be obtained by tuning the model, we felt tha having the same model hyperparameters operate on all the problems would make the main message of the paper stronger
  - All our models used a single layer LSTM with either 256 or 512 hidden units
  - Trained with stochastic gradient descent with a learning rate of 1.0, batch size of 128, random uniform weight initialization from -0.08 to 0.08, and L2 gradient clipping of 2.0
  - Generate 1M training example pairs, and we did observe overfitting in some cases where the task was simpler (i.e., for small n)

Experiment

- **Convex Hull**
  - Accuracy
    - Consider output sequences $\mathcal{C}^1$ & $\mathcal{C}^2$ to be the same if they represent the same polygon
  - Area covered of the true convex hull
    - The output represents a simple polygon (i.e., without self-intersections)
  - If an algorithm fails to produce a simple polygon in more than 1% of the cases we simply reported FAIL

Experiment

- **Convex Hull**
  - The inputs are presented to the encoder during inference affects its performance
  - When the points on the true convex hull are seen “late” in the input sequence, the accuracy is lower
  - This is possibly the network does not have enough processing steps to “update” the convex hull it computed until the latest points were seen

METHOD	TRAINED n	n	ACCURACY	AREA
LSTM [1]	50	50	1.9%	FAIL
+ATTENTION [5]	50	50	38.9%	99.7%
PTR-NET	50	50	72.6%	99.9%
LSTM [1]	5	5	87.7%	99.6%
PTR-NET	5-50	5	92.0%	99.6%
LSTM [1]	10	10	29.9%	FAIL
PTR-NET	5-50	10	87.0%	99.8%
PTR-NET	5-50	50	69.6%	99.9%
PTR-NET	5-50	100	50.3%	99.9%
PTR-NET	5-50	200	22.1%	99.9%
PTR-NET	5-50	500	1.3%	99.2%

Experiment

- **Delaunay Triangulation**
  - TSP arises in many areas of theoretical computer science and is an important algorithm used for microchip design or DNA sequencing
  - Focused on the planar symmetric TSP:
    given a list of cities, we wish to find the shortest possible route that visits each city exactl once and returns to the starting point
  - Assume the distance between two cities is the same in each opposite direction
  - NP-hard problem which allows us to test the capabilities and limitations of our model

Experiment

- **Travelling Salesman Problem**
  - TSP arises in many areas of theoretical computer science and is an important algorithm used for microchip design or DNA sequencing
  - Focused on the planar symmetric TSP:
    given a list of cities, we wish to find the shortest possible route that visits each city exactl once and returns to the starting point
  - Assume the distance between two cities is the same in each opposite direction
  - NP-hard problem which allows us to test the capabilities and limitations of our model

Experiment

- **Conclusions**
  - TSP arises in many areas of theoretical computer science and is an important algorithm used for microchip design or DNA sequencing
  - Focused on the planar symmetric TSP:
    given a list of cities, we wish to find the shortest possible route that visits each city exactl once and returns to the starting point
  - Assume the distance between two cities is the same in each opposite direction
  - NP-hard problem which allows us to test the capabilities and limitations of our model

Part 1.

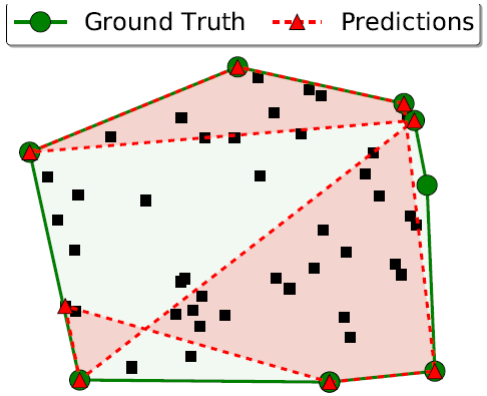
Introduction

Peter Brown et al. The mathematics of statistical machine translation: Parameter estimation. Computational Linguistics, 19(2):263–311, 1993.

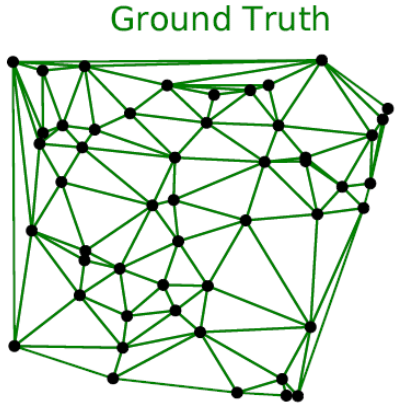
Introduction

80.7% m $n = 5$

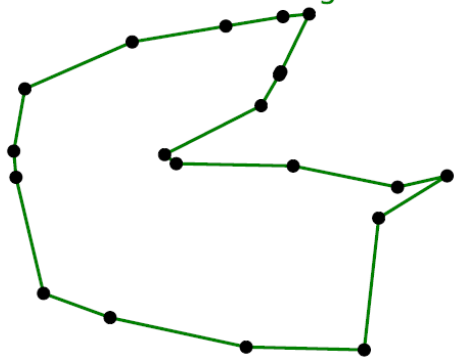
93.0% $n = 10$



(a) LSTM, m=50, n=50



(b) Truth, n=50

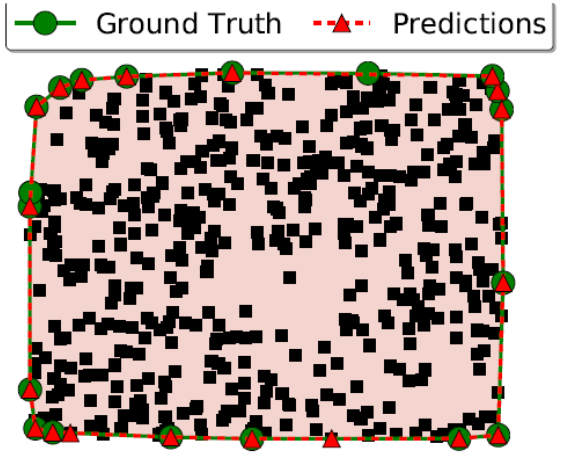


(c) Truth, n=20

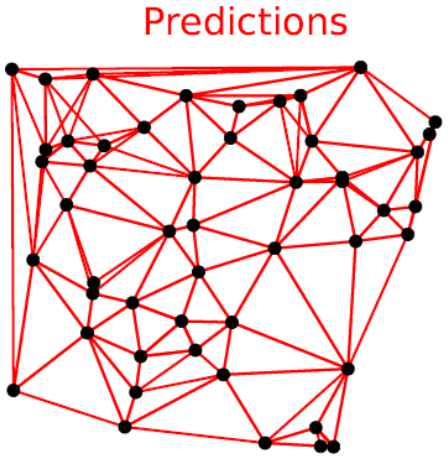
22.6% $n = 50$

52.8% $O(n^2)$

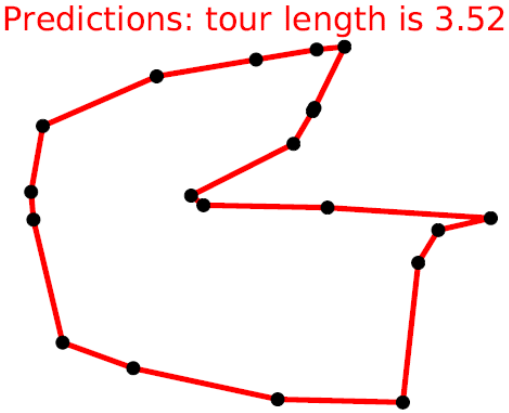
81.3%



(d) Ptr-Net, m=5-50, n=500



(e) Ptr-Net , m=50, n=50



(f) Ptr-Net , m=5-20, n=20

Peter Brown et al. The mathematics of statistical machine translation: Parameter estimation. Computational Linguistics, 19(2):263–311, 1993.

Part 1.

Introduction

$\upsilon\, W_1\, W_2\, u^i \quad j \in (1, \ldots, n) \quad P_{C_{i-1}} \quad (e_1, \ldots, e_n)$

$\upsilon \quad n \quad (\mathcal{P}, \mathcal{C}^{\mathcal{P}}) \quad (d_1, \ldots, d_{m(\mathcal{P})})$

$\Rightarrow \Leftarrow$

$O(n \log n)$

$1 \quad \mathcal{C}^{\mathcal{P}}$

$C_i = (1, 2, 4)$ $O(n^3)$ (2,4,1) $O(n^2)$ $n = 20$ $1\% 100\%$ $n = 500$ FAIL

Ptr-Net “late” “update” “pointing” $n = 50$ $n = 500$ **w.r.t** n' $n' \neq n$

Peter Brown et al. The mathematics of statistical machine translation: Parameter estimation. Computational Linguistics, 19(2):263–311, 1993.

Introduction

n	OPTIMAL	A1	A2	A3	PTR-NET
5	2.12	2.18	2.12	2.12	2.12
10	2.87	3.07	2.87	2.87	2.88
50 (A1 TRAINED)	N/A	6.46	5.84	5.79	6.42
50 (A3 TRAINED)	N/A	6.46	5.84	5.79	6.09
5 (5-20 TRAINED)	2.12	2.18	2.12	2.12	2.12
10 (5-20 TRAINED)	2.87	3.07	2.87	2.87	2.87
20 (5-20 TRAINED)	3.83	4.24	3.86	3.85	3.88
25 (5-20 TRAINED)	N/A	4.71	4.27	4.24	4.30
30 (5-20 TRAINED)	N/A	5.11	4.63	4.60	4.72
40 (5-20 TRAINED)	N/A	5.82	5.27	5.23	5.91
50 (5-20 TRAINED)	N/A	6.46	5.84	5.79	7.66

$n > 20$ 10% $n = 50$

n $n = 30$ 20 40

$O(n \log n)$ $O(n \log n)$ $n = 25$

Peter Brown et al. The mathematics of statistical machine translation: Parameter estimation. Computational Linguistics, 19(2):263–311, 1993.