Effective Attention Mechanisms for Sequence Learning

LI, Jian

Ph.D. Oral Defense

Supervisor: Prof. Michael R. Lyu

2020/06/16

香港中文大學
The Chinese University of Hong Kong

Sequential Data is Prevalent

It was the best of times, it was the worst of times, it was the age of wisdom, it was the age of foolishness...

Sequence Learning

- Discover valuable knowledge from sequential data.
  - E.g., forecasting
  - Extracting the patterns

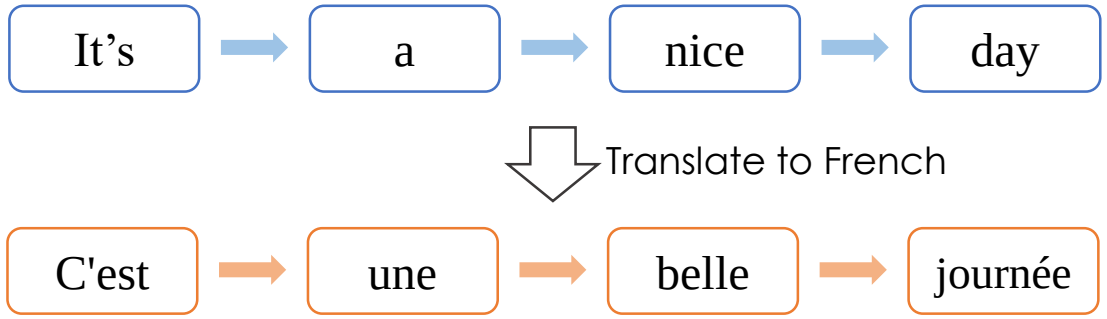
- Manual analysis is inefficient and error-prone.
- Sequence learning automatically finds statistically relevant patterns.

Sequence Learning Tasks

- Sequence Prediction

(Sentence Completion)

- Sequence Generation



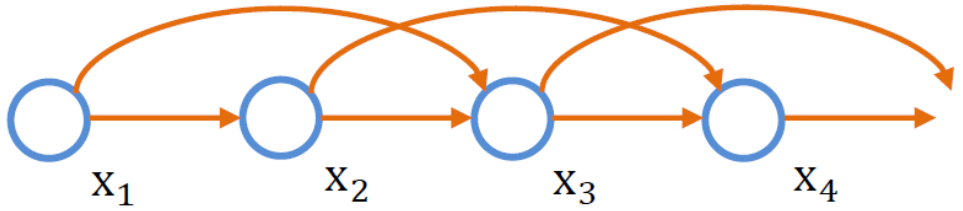
(Machine Translation)

The core problem of sequence learning is *dependency modeling*.

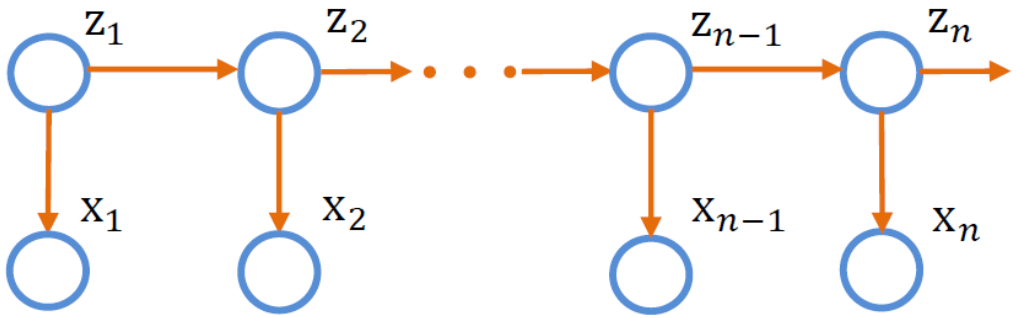
[Agrawal et al., *Mining Sequential Patterns,* 1995]

Traditional Sequence Learning

- Markov Models:



(a) second-order Markov chain

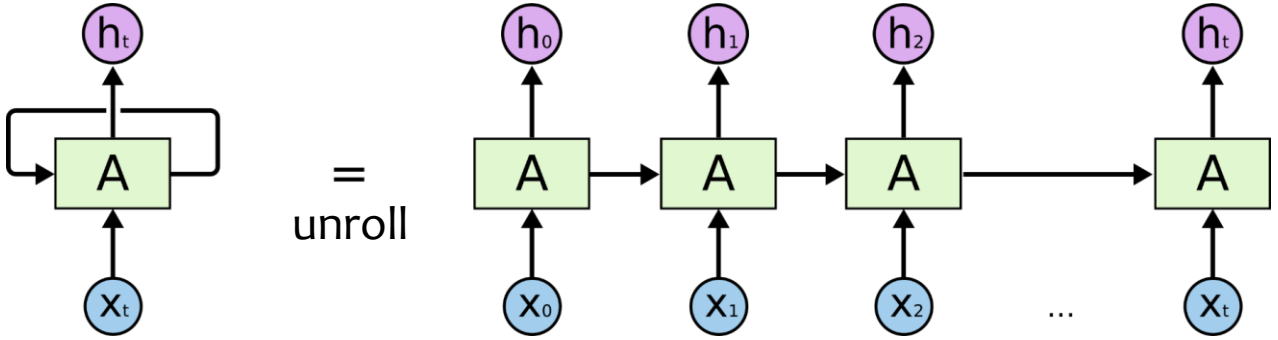


(b) hidden Markov model

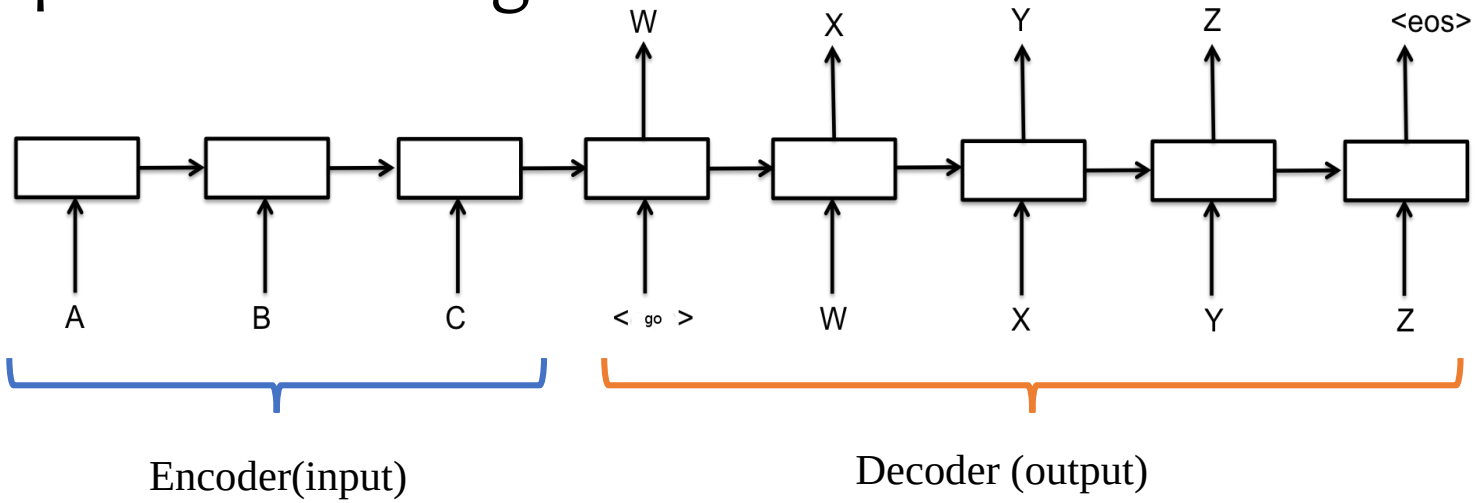
  - Fail to capture **long-term** dependencies

Neural Sequence Learning

- Recurrent Neural Networks (RNNs)



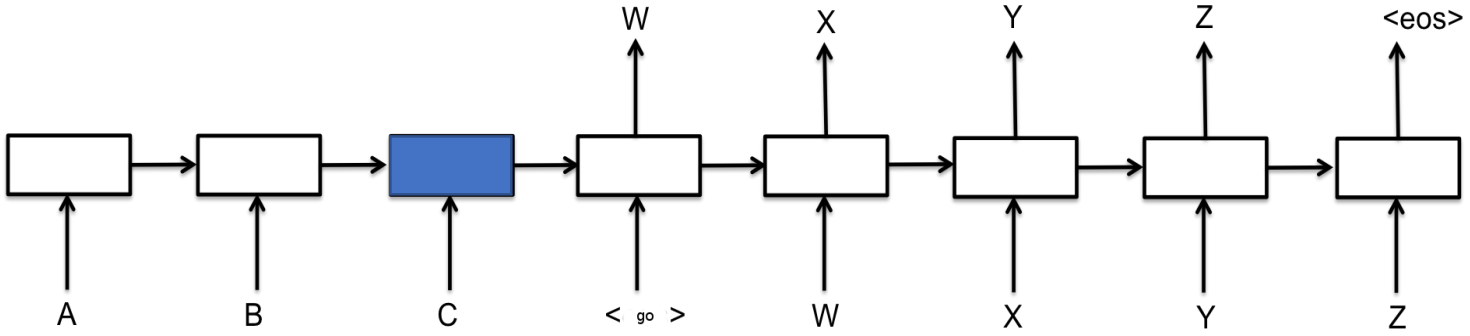
- Sequence-to-Sequence Learning



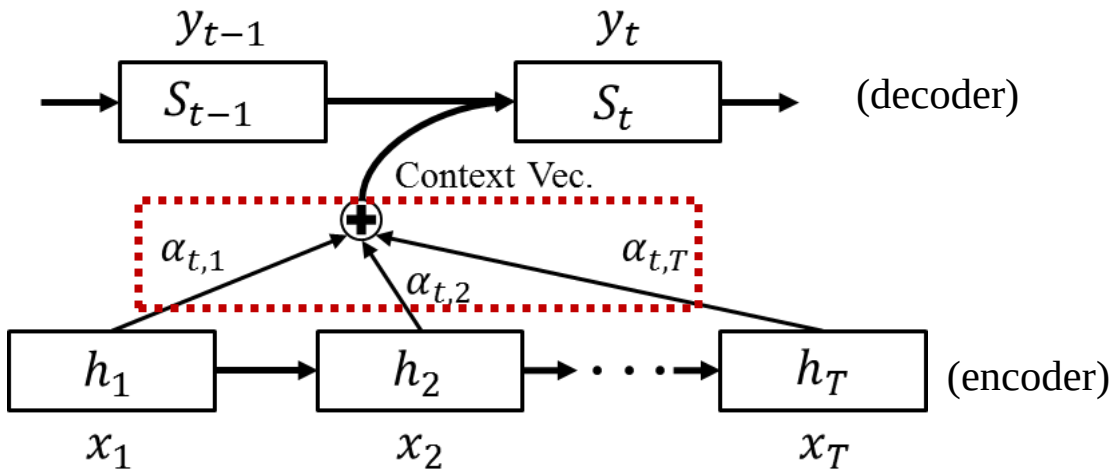
https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Attention Mechanism

- Hidden state bottleneck of RNN:
  - The source sequence is encoded in one fixed-size vector.



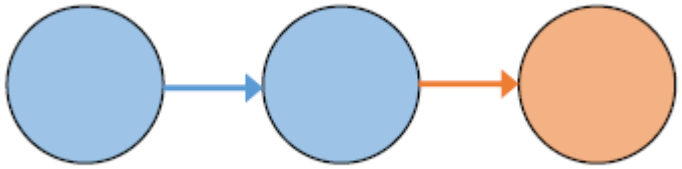
✓Attention adds *shortcut connections* to all source elements.



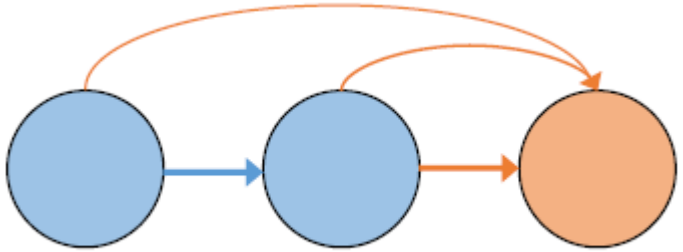
- Attention weights (connection strengths)
- Context vector (weighted summation)

Self-Attention Mechanism

- RNN's sequential nature hinders **parallel computation.**



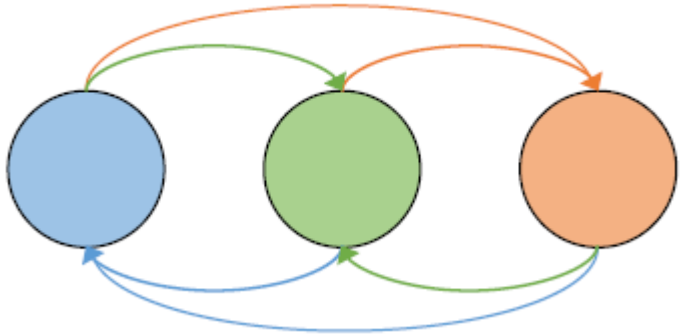
RNN



RNN with Attention

Shallow Attention

- Self-Attention Networks (SANs):
  - Discard recurrent architectures
  - Rely solely on attention mechanisms
  - Simultaneously capture dependencies among all elements
  - Positional encoding to record order information



SAN

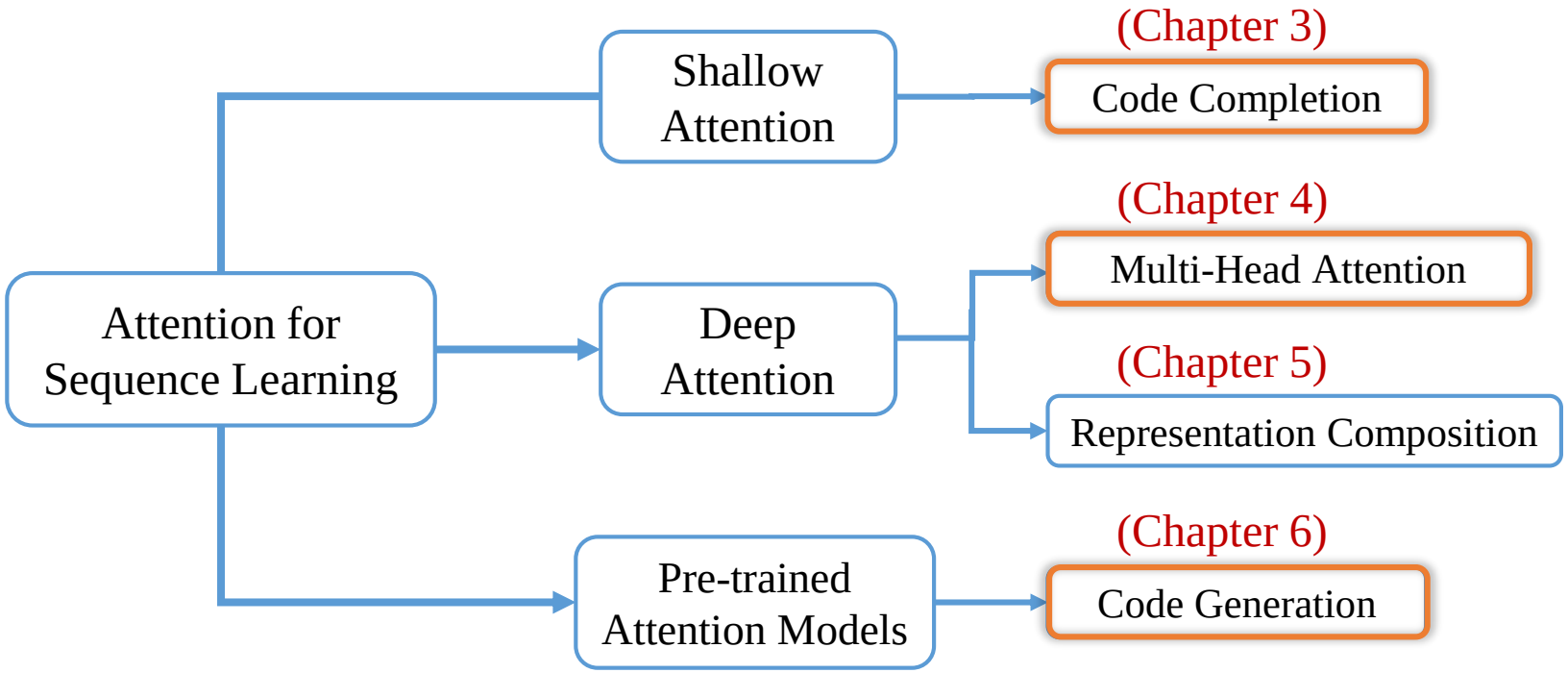
Deep Attention

Challenges for Attention Mechanisms

- Application domains other than text
  - Source code data
  - Highly structured, large vocabulary

- Model design deficiencies
  - Deep self-attention involves multiple attention heads/layers.
  - How to coordinate these components?

Thesis Contributions



[IJCAI'18]

[EMNLP'18, NAACL'19]

[AAAI'20]

[*CIKM'20]

* In Submission

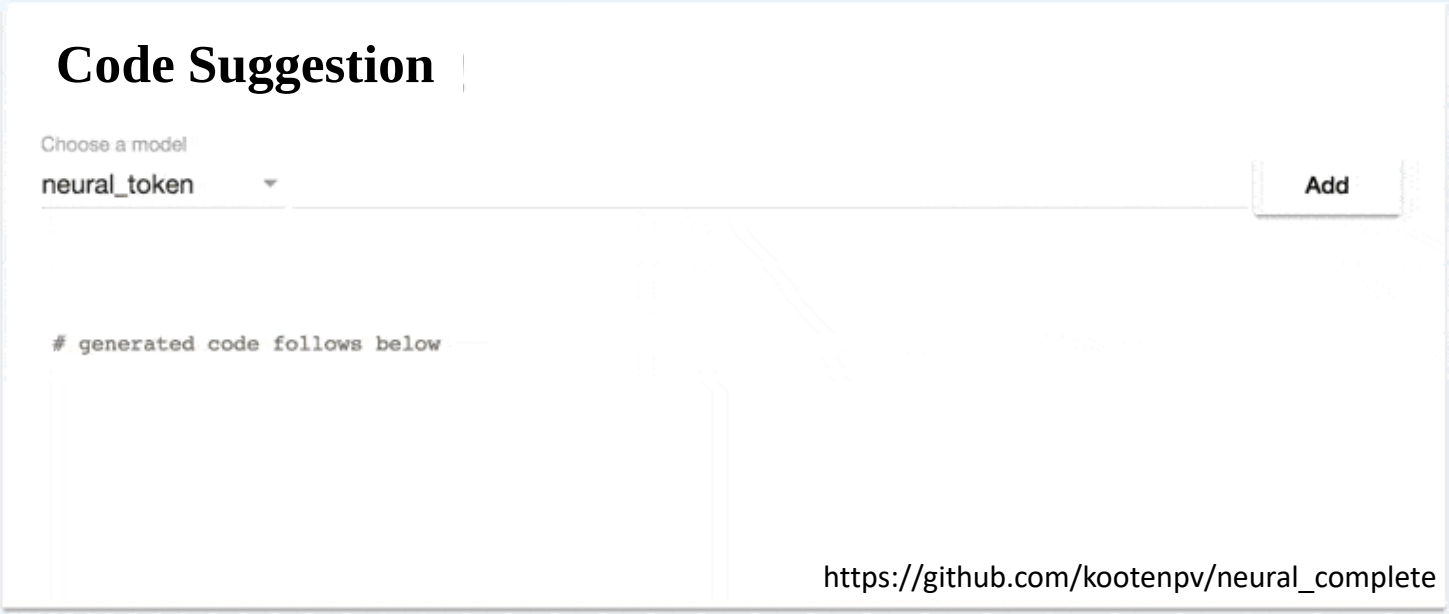
Outline

- Topic 1: Neural Attention for Code Completion
- Topic 2: Multi-Head Self-Attention
- Topic 3: Pre-trained Attention for Code Generation
- Conclusion and Future Work

Outline

- Topic 1: Neural Attention for Code Completion
- Topic 2: Multi-Head Self-Attention
- Topic 3: Pre-trained Attention for Code Generation
- Conclusion and Future Work

Code Completion



- Static programming language: C++, JAVA
  - compile-time type information
- Dynamic programming language: Python, JavaScript
  - learning-based language models

Code Completion with Language Models

- Simplified problem:
  - given a sequence of code tokens, we predict the next one token.

$$p(w_t|w_1, w_2, \ldots, w_{t-1}; \theta)$$

- Method: adapt neural language models (e.g. RNNs) for code completion.

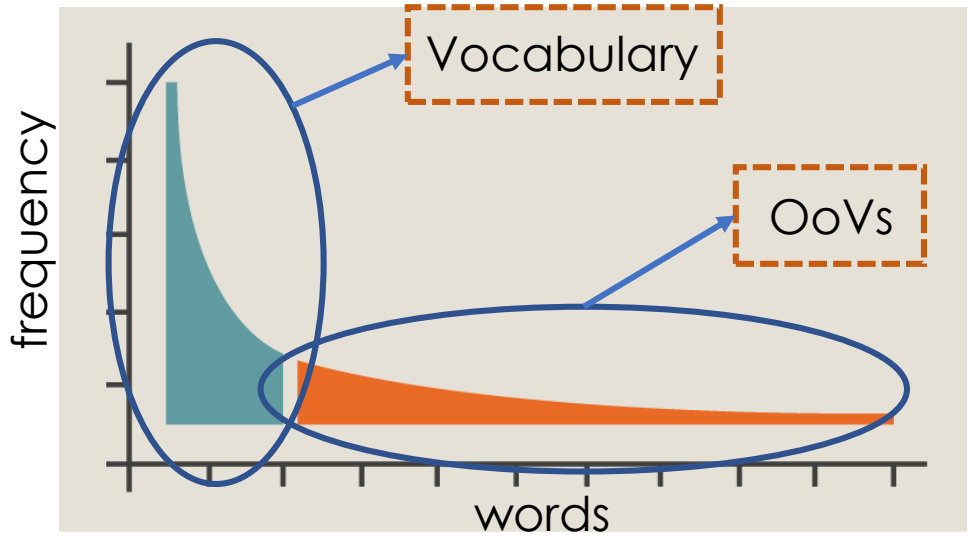
Challenges

1. Long-range dependencies.

```
// JavaScript source code

// Set Up Class
class Horse {
    constructor(name, trainer) {
        this._name = name;
        this._trainer = trainer;
    }

    get name() {
        return this._name;
    }

    get trainer() {
        return this._trainer;
    }
}

// Define Variable
txt = "";

    :

// Create Instances
myHorse1 = new Horse('Spirit of Wedza', 'Julie Camacho');
myHorse2 = new Horse('True North', 'Mark Prescott');

// Increment txt
txt += myHorse1.name + " is trained by " + myHorse1.trainer + "<br>";
txt += myHorse2.name + " is trained by " + myHorse2.trainer + "<br>";
```

Challenges

2. Out-of-Vocabulary (OoV) words.
   - Rare words
   - User-defined identifiers



OoVs cannot be correctly predicted!

Methods

- Deal with long-range dependencies:
  - ✓Abstract Syntax Tree (AST)

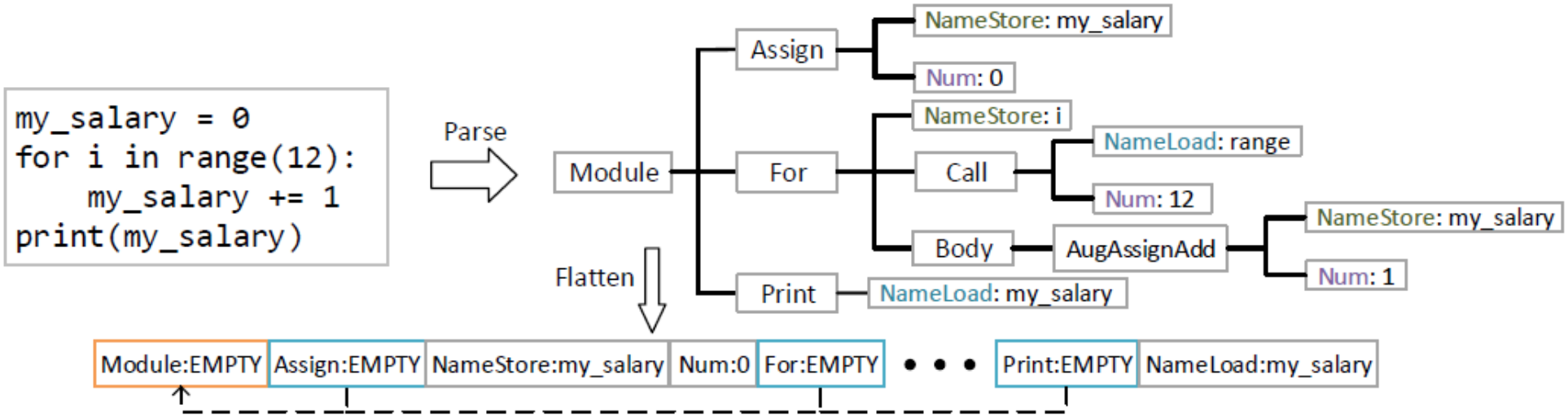


Figure 1: A Python program and its corresponding abstract syntax tree (AST). The dashed arrow points to a parent node.

Problem: given a sequence of AST nodes, predict the next one AST node, including *type* and *value*.

Methods

- Deal with long-range dependencies:
  - ✓Abstract Syntax Tree (AST)

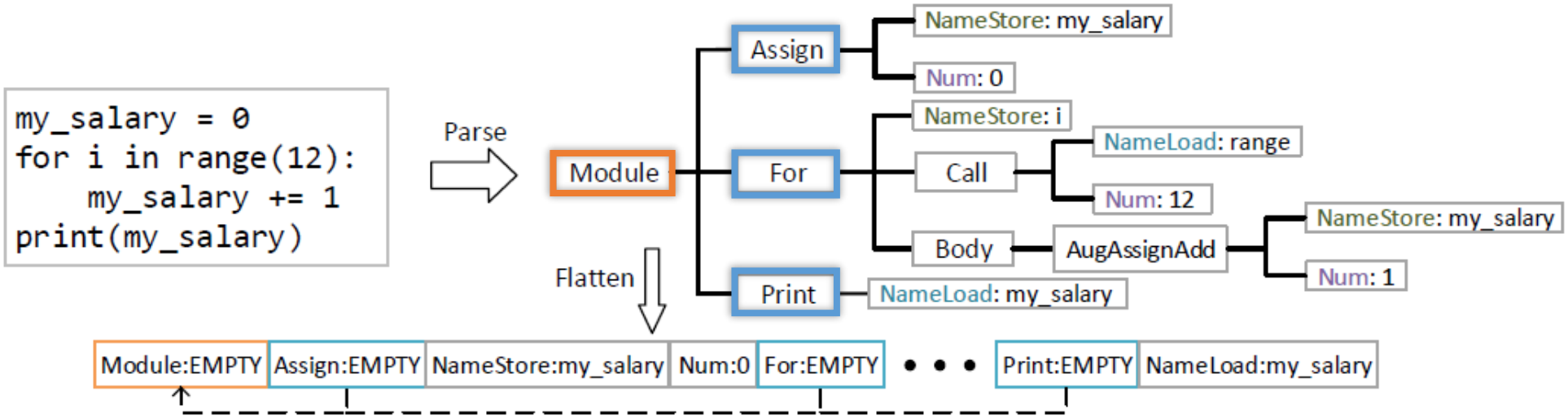
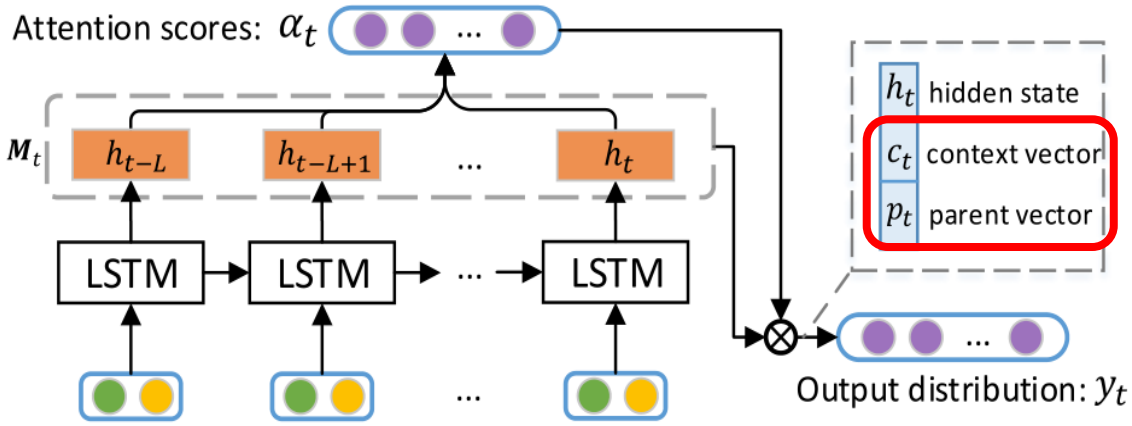


Figure 1: A Python program and its corresponding abstract syntax tree (AST). The dashed arrow points to a parent node.

Exploit the parent-children information on program's AST.

Attention Mechanisms

- Deal with long-range dependencies:
  - ✓Parent attention



$$A_t = v^T \tanh(W^m M_t + (W^h h_t) 1_L^T)$$
$$\alpha_t = softmax(A_t)$$
$$c_t = M_t \alpha_t^T$$
$$G_t = \tanh(W^g [h_t; c_t; p_t])$$
$$y_t = softmax(W^v G_t + b^v)$$

p_t is the parent vector storing hidden state of the parent node on AST.

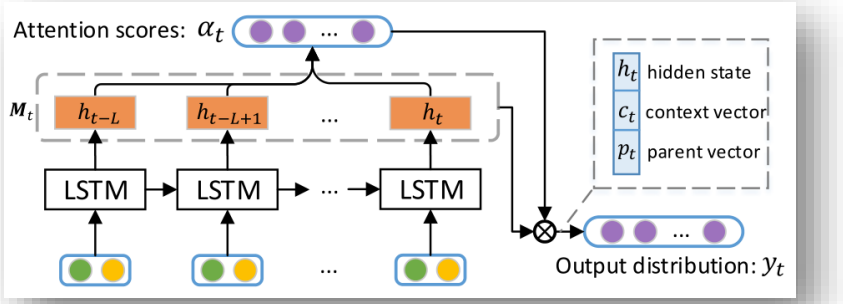
Methods

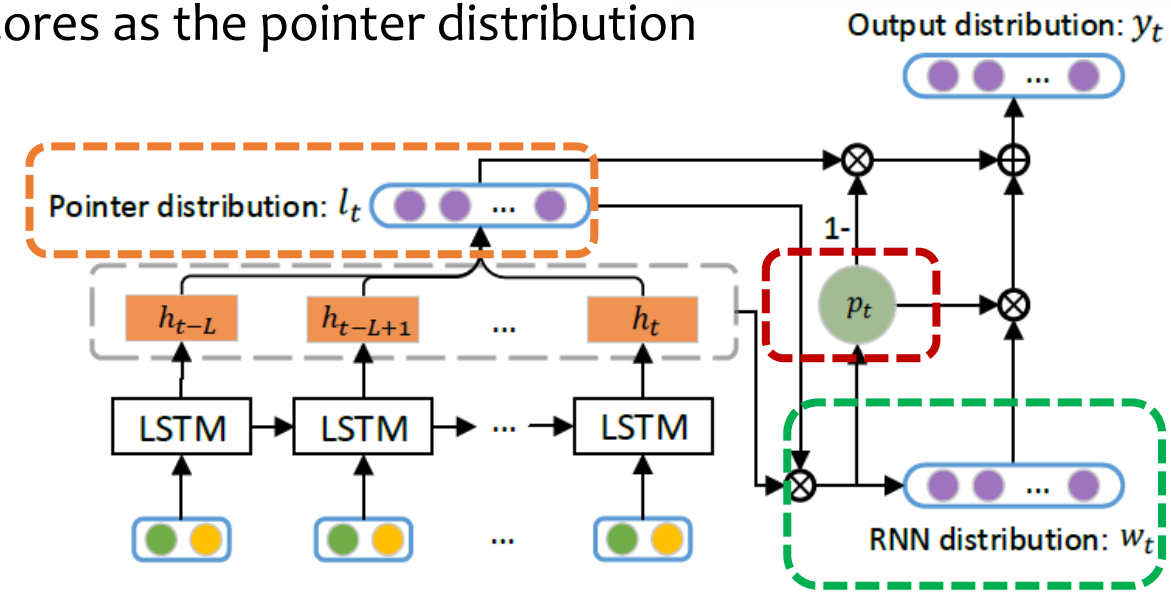
- Deal with OoV words:
  - ✓Locally repeated terms are prevalent.
  - ✓Intuition: copy from local context to predict OoVs. (Pointer Network)

```
<script>

    var grid_length = 75;
    // Note that grid_length in the book is 100.
    // I reduced it here for smooth performance on mobile devices.
    var grid = [];
    var temp_grid = [];
    var beta = 0.05;
    var gamma = 0.15;

    function get_random_int(min, max) {
        return Math.floor(Math.random() * (max - min + 1)) + min;
    }

    function init_grid() {
        for (var i = 0; i < grid_length; i = i + 1) {
            grid[i] = [];
            for (var ii = 0; ii < grid_length; ii = ii + 1) {
               grid[i][ii] = "S";
            }
        }
        grid[get_random_int(0,grid_length-1)][get_random_int(0,grid_length-1)] = "I";
    }

    init_grid();

    draw_grid(grid,["S","#dcdcdc","I","#c82605","R","#6fc041"]);
```

Pointer Mixture Network

- Deal with OoV words:
  - ➢ Global RNN component
  - ➢ Local pointer component
    - ❑ Reuse the attention scores as the pointer distribution
  - ➢ Controller





$$p_t = \sigma(W^p[h_t; c_t] + b^p)$$
$$y_t = softmax([p_t w_t; (1 - p_t) l_t])$$

Learns *when* and *where* to copy.

Experiments

- Datasets:
  - JavaScript (JS) and Python (PY)
  - Type prediction and value prediction.

	JavaScript	Python
Training Queries	$10.7 * 10^7$	$6.2 * 10^7$
Test Queries	$5.3 * 10^7$	$3.0 * 10^7$
Type Vocabulary	95	329
Value Vocabulary	$2.6 * 10^6$	$3.4 * 10^6$

OoV problem!

Experiments

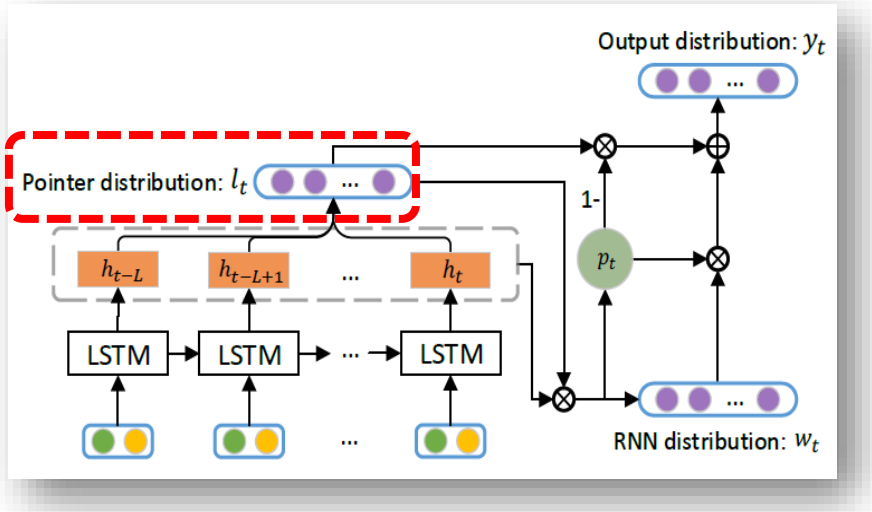
- Accuracies on next value prediction with different vocabulary sizes

Vocab. Size	JS (1 K)	JS (10 K)	JS (50 K)	PY (1 K)	PY (10 K)	PY (50 K)
OoV Rate	20%	11%	7%	24%	16%	11%
Vanilla LSTM	69.9%	75.8%	78.6%	63.6%	66.3%	67.3%
Attentional LSTM (Ours)	71.7%	78.1%	80.6%	64.9%	68.4%	69.8%
Pointer Mixture Network (Ours)	**73.2%**	**78.9%**	**81.0%**	**66.4%**	**68.9%**	**70.1%**

- Vanilla LSTM: without attention nor pointer.
- Attentional LSTM: context attention and parent attention.
- Pointer Mixture Network: both attention and pointer.

Experiments

- Is the learned pointer distribution meaningful?
  - Pointer Random Network: randomize the pointer distribution

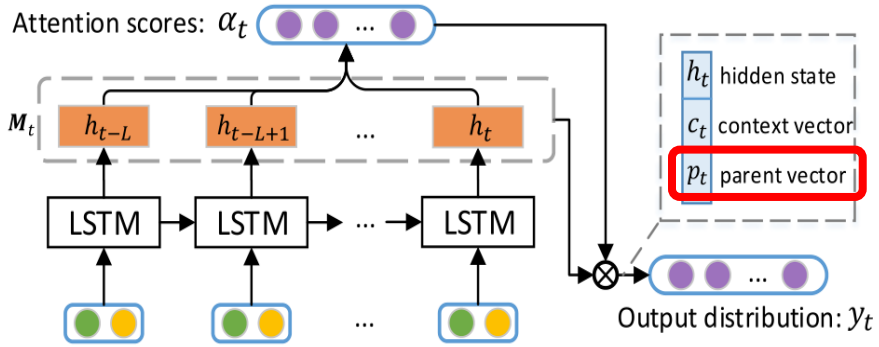


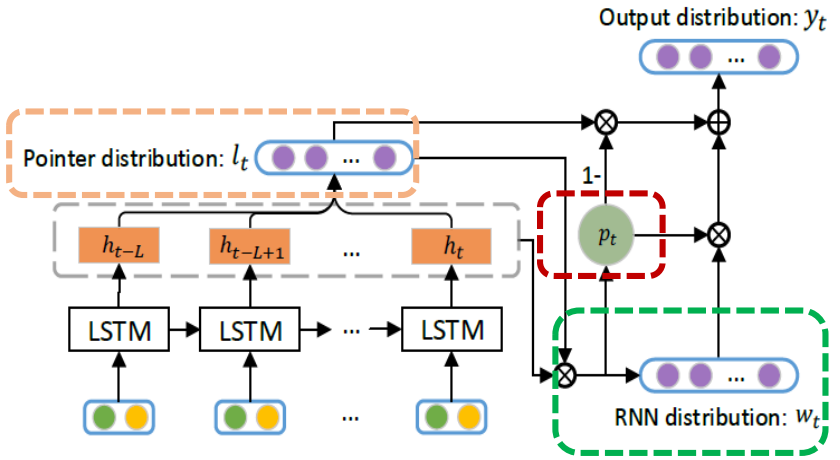
	JS_1k	PY_1k
Pointer Random Network	71.4%	64.8%
Attentional LSTM	71.7%	64.9%
Pointer Mixture Network	**73.2%**	**66.4%**

Pointer mixture network indeed learns when and where to copy OoVs.

Summary

1. Propose a parent attention mechanism for AST-based code completion.
2. Propose a pointer mixture network which learns to either generate a new value or copy an OoV value from local context.
3. Demonstrate the effectiveness of our model via experiments.





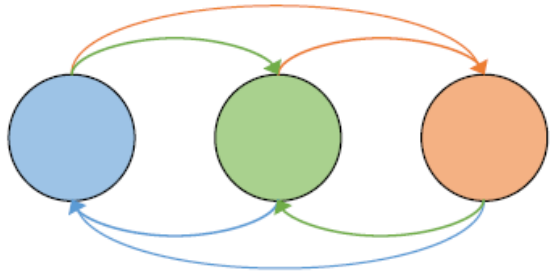
Outline

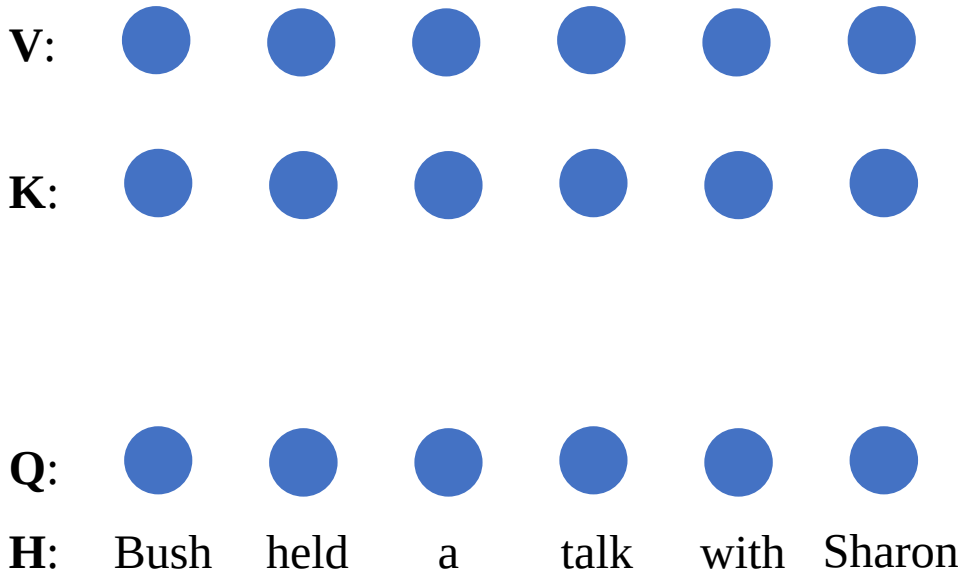
- Topic 1: Neural Attention for Code Completion
- Topic 2: Multi-Head Self-Attention
- Topic 3: Pre-trained Attention for Code Generation
- Conclusion and Future Work

Self-Attention Mechanism

Linear Transformation

$$\begin{bmatrix}\mathbf{Q}\\ \mathbf{K}\\ \mathbf{V}\end{bmatrix} = \mathbf{H}\begin{bmatrix}\mathbf{W}_Q\\ \mathbf{W}_K\\ \mathbf{W}_V\end{bmatrix}$$





[Vaswani et al., *Attention is All You Need,* 2017]

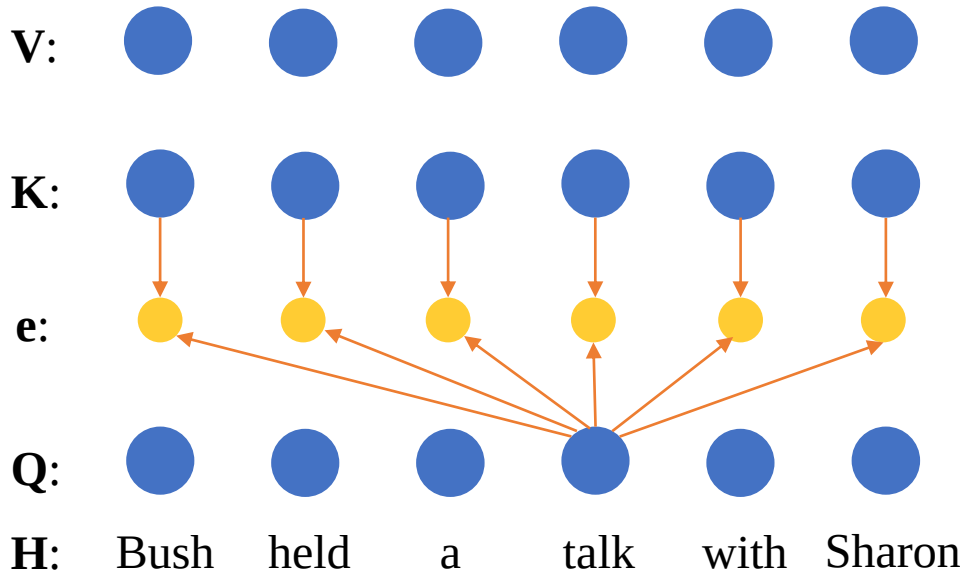
Self-Attention Mechanism

Linear Transformation

$$\begin{bmatrix}\mathbf{Q}\\ \mathbf{K}\\ \mathbf{V}\end{bmatrix} = \mathbf{H}\begin{bmatrix}\mathbf{W}_Q\\ \mathbf{W}_K\\ \mathbf{W}_V\end{bmatrix}$$

Attention Weights

$$\text{Att}(\mathbf{Q},\mathbf{K}) = softmax(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d}})$$



[Vaswani et al., *Attention is All You Need,* 2017]

Self-Attention Mechanism

Linear Transformation

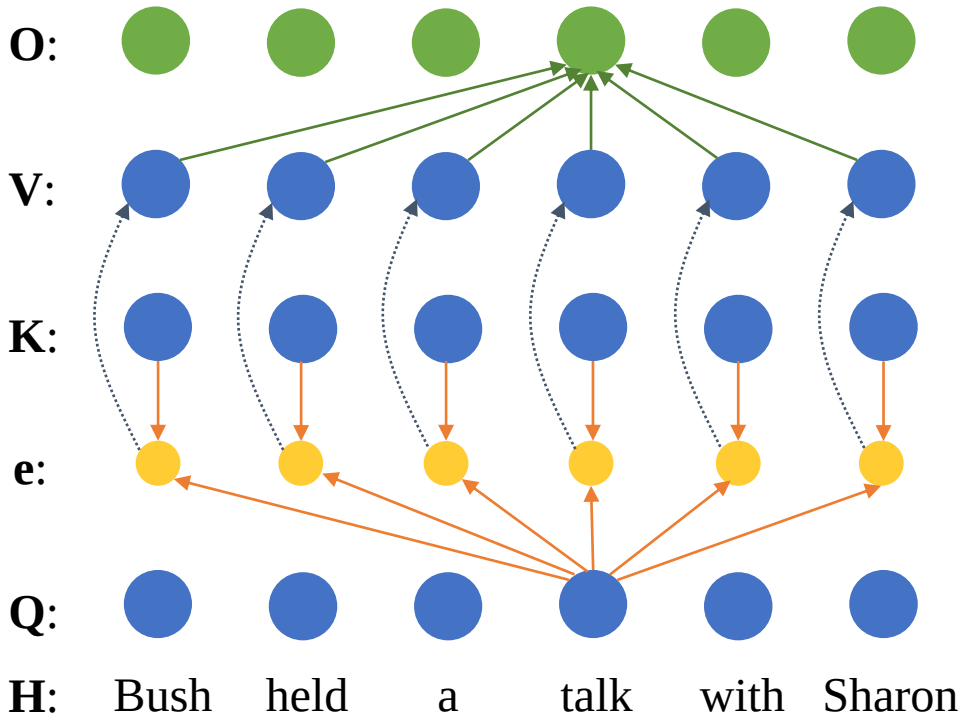
$$\begin{bmatrix}\mathbf{Q}\\ \mathbf{K}\\ \mathbf{V}\end{bmatrix} = \mathbf{H}\begin{bmatrix}\mathbf{W}_Q\\ \mathbf{W}_K\\ \mathbf{W}_V\end{bmatrix}$$

Attention Weights

$$\mathrm{Att}(\mathbf{Q},\mathbf{K}) = softmax(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d}})$$

Weighted Sum

$$\mathbf{O} = \mathrm{Att}(\mathbf{Q},\mathbf{K})\cdot\mathbf{V}$$



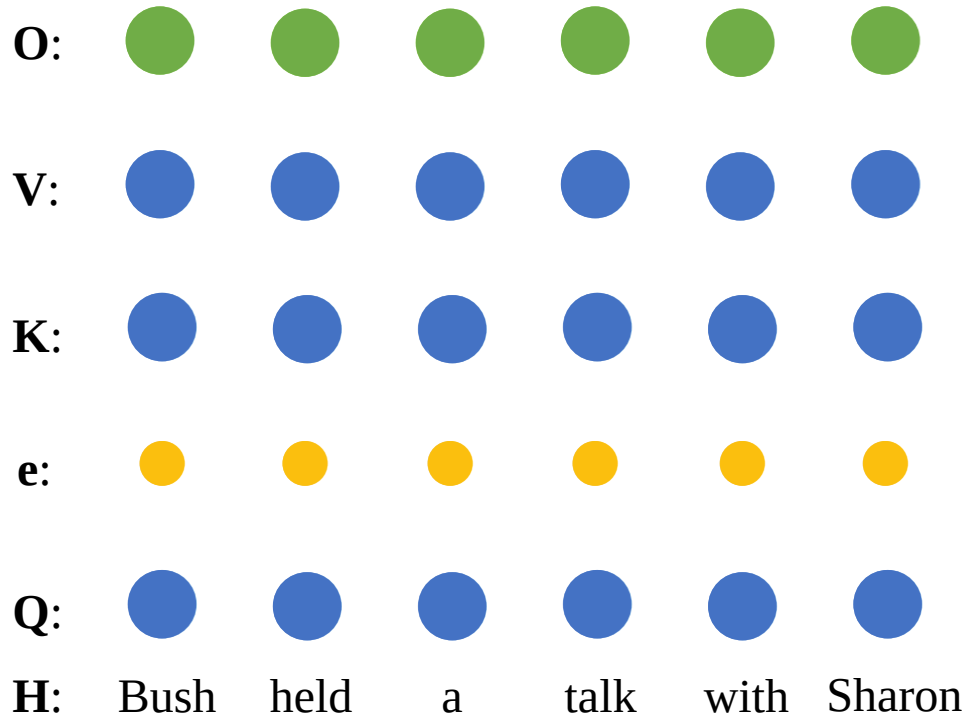
[Vaswani et al., *Attention is All You Need,* 2017]

Multi-Head Self-Attention

$$\begin{bmatrix} \mathbf{Q}^h \\ \mathbf{K}^h \\ \mathbf{V}^h \end{bmatrix} = \mathbf{H} \begin{bmatrix} \mathbf{W}_Q{}^h \\ \mathbf{W}_K{}^h \\ \mathbf{W}_V{}^h \end{bmatrix}$$

$$\mathrm{Att}(\mathbf{Q}^h, \mathbf{K}^h) = softmax(\frac{\mathbf{Q}^h \mathbf{K}^{hT}}{\sqrt{d_k}})$$

$$\mathbf{O}^h = \mathrm{Att}(\mathbf{Q}^h, \mathbf{K}^h) \cdot \mathbf{V}^h$$



[Vaswani et al., *Attention is All You Need,* 2017]

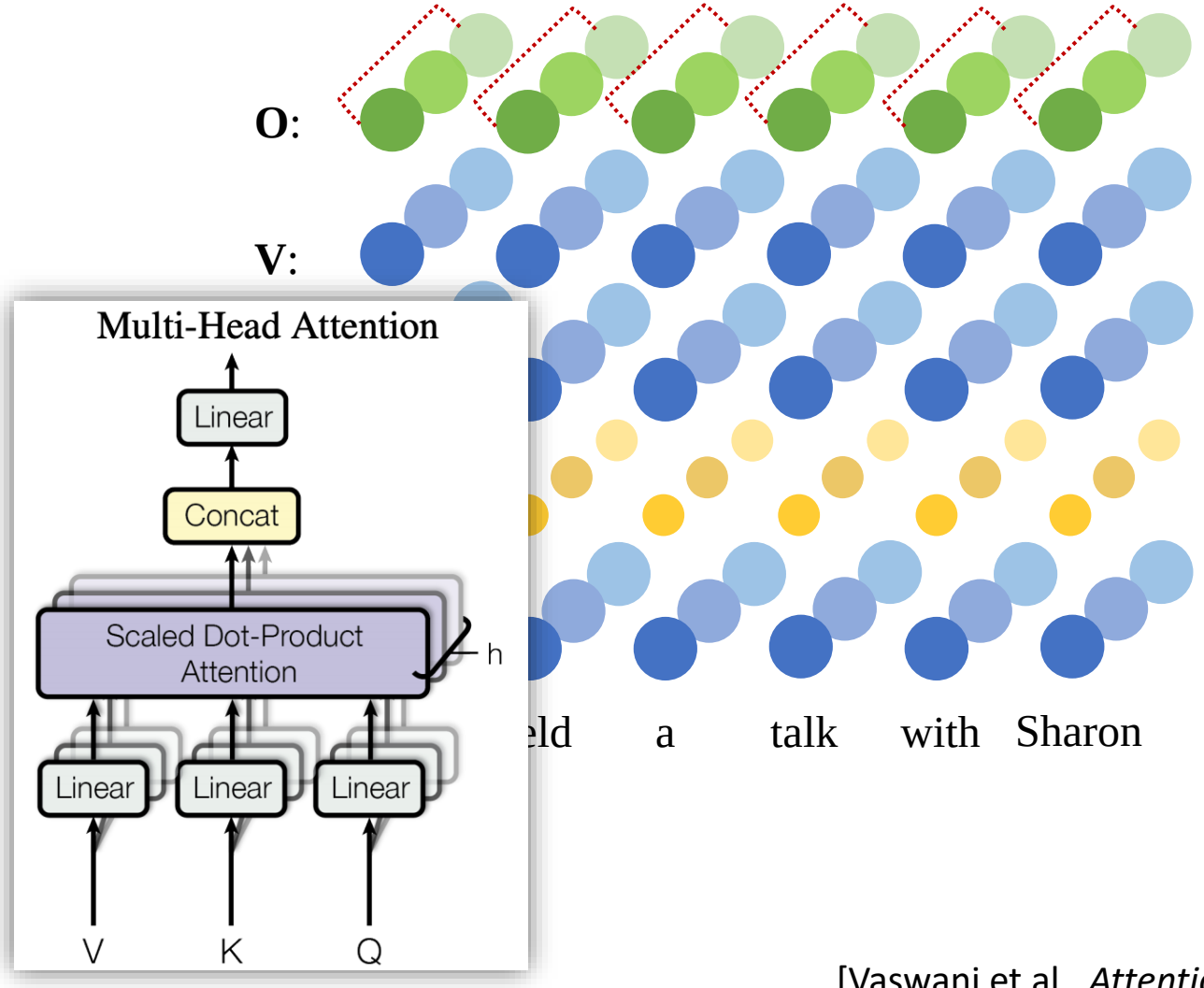
Multi-Head Self-Attention

$$\begin{bmatrix} \mathbf{Q}^h \\ \mathbf{K}^h \\ \mathbf{V}^h \end{bmatrix} = \mathbf{H} \begin{bmatrix} \mathbf{W}_Q{}^h \\ \mathbf{W}_K{}^h \\ \mathbf{W}_V{}^h \end{bmatrix}$$

$$\text{Att}(\mathbf{Q}^h, \mathbf{K}^h) = softmax(\frac{\mathbf{Q}^h \mathbf{K}^{hT}}{\sqrt{d_k}})$$

$$\mathbf{O}^h = \text{Att}(\mathbf{Q}^h, \mathbf{K}^h) \cdot \mathbf{V}^h$$

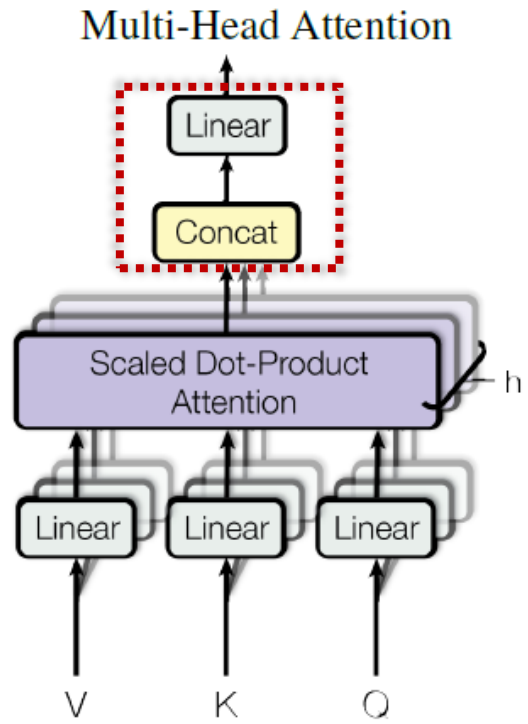
$$\mathbf{O}_f = Concat[\mathbf{O}^1, \dots, \mathbf{O}^H]\mathbf{W}_O$$



[Vaswani et al., *Attention is All You Need,* 2017]

Deficiencies in Multi-Head Attention

- Diversity: jointly extract information from different representation subspaces at different positions.



1. There is no mechanism to guarantee that different attention heads indeed capture distinct information.

   -- Information Extraction

2. The "Concat+Linear" is not expressive enough to aggregate the diverse sub-representations.

   -- Information Aggregation

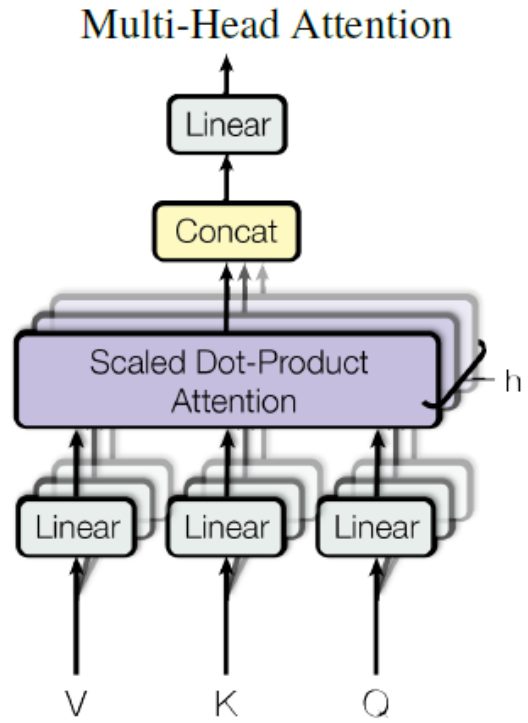
Our Solutions

1. We introduce a disagreement regularization to explicitly encourage the diversity.

   -- Information Extraction

2. We replace the standard linear transformation with an advanced aggregation function.

   -- Information Aggregation



Disagreement Regularization

- Revise the training objective for seq2seq learning (x -> y):

$$J(\theta) = \arg\max_{\theta} \{ \underbrace{L(\mathbf{y}|\mathbf{x};\theta)}_{likelihood} + \lambda * \underbrace{D(\mathbf{a}|\mathbf{x},\mathbf{y};\theta)}_{disagreement} \}$$

  - The auxiliary regularization term $D(\cdot)$ enlarges the distances among multiple attention heads.
  - Do not introduce any new parameters.

Three Types of Disagreement

- *Disagreement on Subspaces* that maximizes the cosine distance among the projected values:

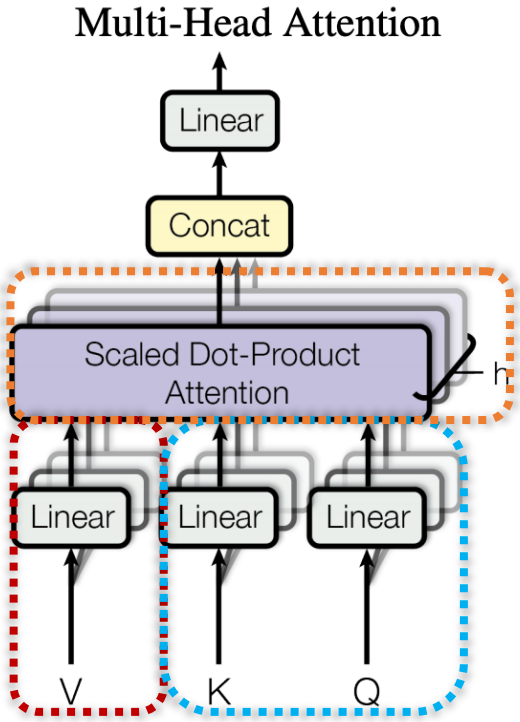
$$D_{subpace} = -\frac{1}{H^2}\sum_{i=1}^{H}\sum_{i=1}^{H}\frac{V^i \cdot V^j}{\|V^i\|\|V^j\|}.$$

- *Disagreement on Positions* that disperses the attended positions predicted by different heads:

$$D_{position} = -\frac{1}{H^2}\sum_{i=1}^{H}\sum_{j=1}^{H}|A^i \odot A^j|. \quad A^h = softmax(\frac{Q^h K^{h^T}}{\sqrt{d_k}})$$

- *Disagreement on Outputs* that maximizes the cosine distance among the outputs of multiple heads:

$$D_{output} = -\frac{1}{H^2}\sum_{i=1}^{H}\sum_{j=1}^{H}\frac{O^i \cdot O^j}{\|O^i\|\|O^j\|}$$

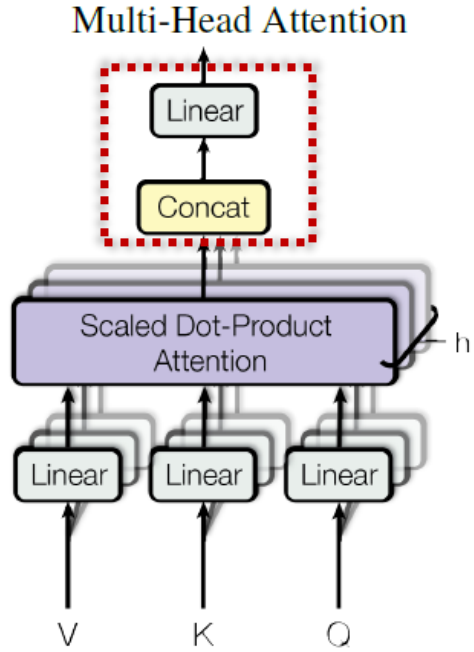


Advanced Aggregation Function

- Linear transformation is a suboptimal feature fusion approach in multi-modal research [1].

We borrow the idea of routing-by-agreement from Capsule Networks [2,3].

  - Iteratively update the proportion of how much a part should be assigned to a whole.



[1] Fukui et al., Multimodal Compact Bilinear Pooling, in EMNLP 2016.
[2] Sabour et al., Dynamic Routing Between Capsules, in NIPS 2017.
[3] Hinton et al., Matrix Capsules with EM Routing, in ICLR 2017.

Routing-by-Agreement

- The information of *H* input capsules is dynamically routed to *N* output capsules.

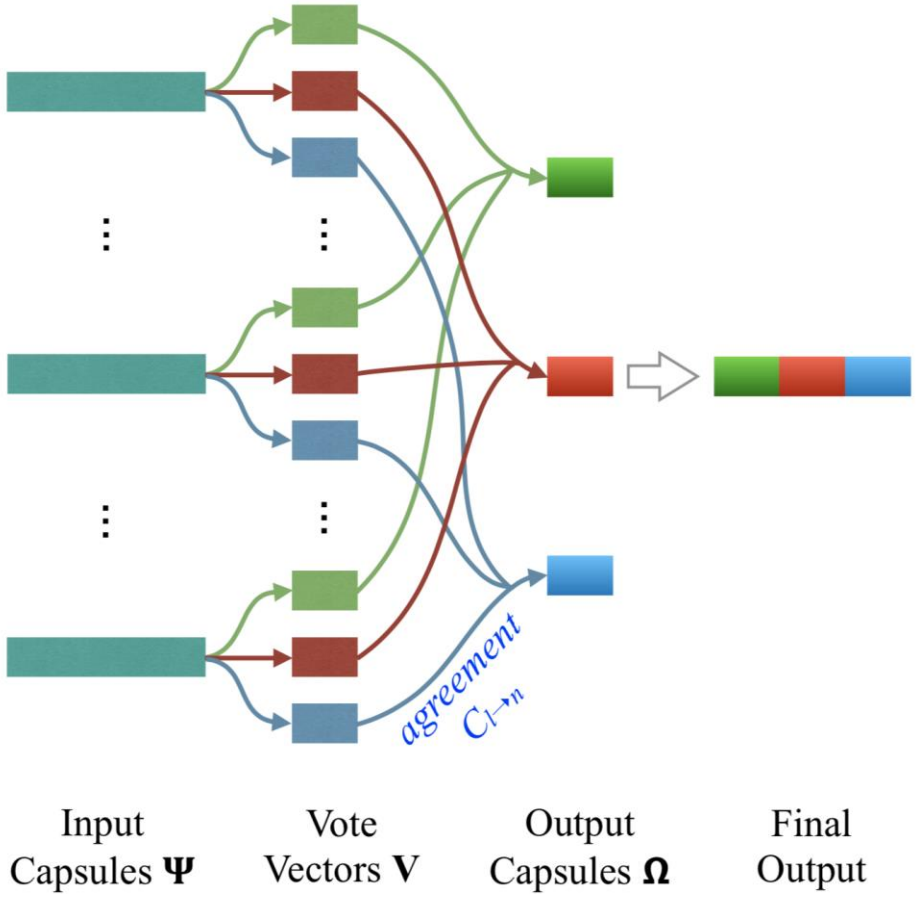
$$\mathbf{\Omega}_h^{in} = f_h(\widehat{\mathbf{O}})$$ Input Capsules

$$\mathbf{V}_{h\to n} = \mathbf{\Omega}_h^{in}\mathbf{W}_{h\to n}$$ Vote Vectors

$$\mathbf{\Omega}_n^{out} = \frac{\sum_{h=1}^{H} C_{h\to n}\mathbf{V}_{h\to n}}{\sum_{h=1}^{H} C_{h\to n}}$$ Output Capsules

- Concatenate *N* output capsules to form the final.

$$\mathbf{O} = [\mathbf{\Omega}_1^{out}, \ldots, \mathbf{\Omega}_N^{out}]$$



Routing-by-Agreement

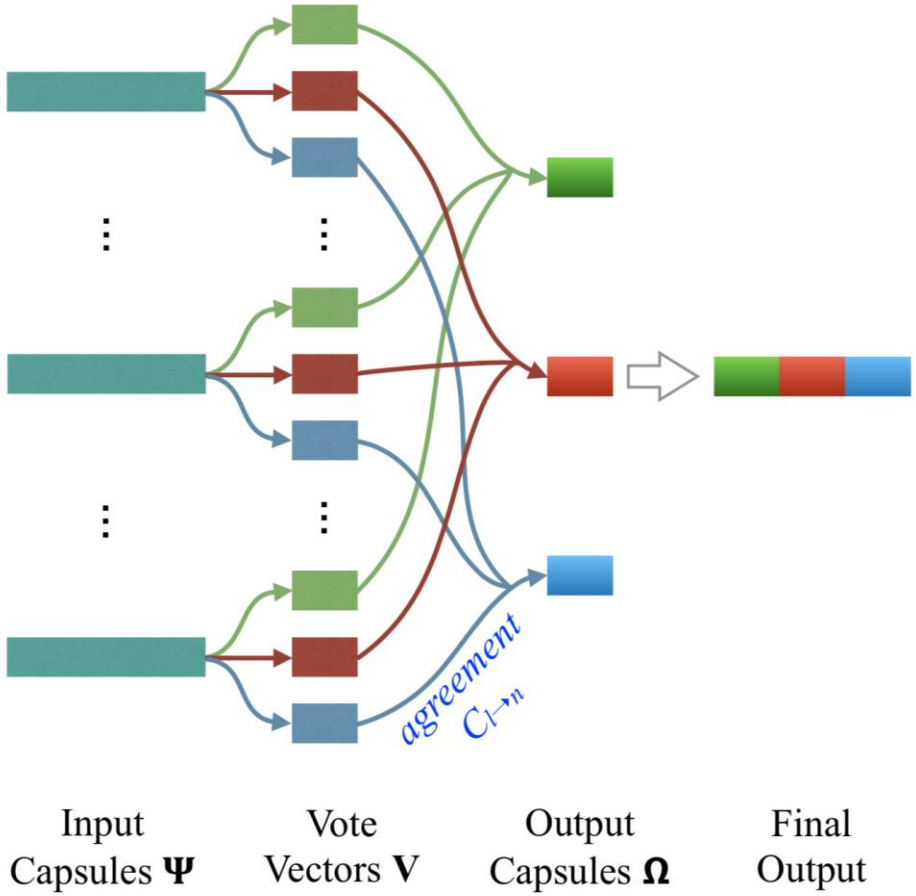
- Two representative routing algorithms for $C_{h \to n}$:

Algorithm 1 Iterative Simple Routing.

procedure ROUTING($\mathbf{V}, T$):
  $\forall \mathbf{V}_{h \to *}$: $B_{h \to n} = 0$
  for T iterations **do**
    $\forall \mathbf{V}_{h \to *}$: $C_{h \to n} = \frac{\exp(B_{h \to n})}{\sum_{n'=1}^{N} \exp(B_{h \to n'})}$
    $\forall \mathbf{\Omega}_n^{out}$: compute $\mathbf{\Omega}_n^{out}$ by Eq. 7
    $\forall \mathbf{V}_{h \to *}$: $B_{h \to n} \mathrel{+}= \mathbf{\Omega}_n^{out} \cdot \mathbf{V}_{h \to n}$
  return $\mathbf{\Omega}$

Algorithm 2 Iterative EM Routing.

procedure EM ROUTING($\mathbf{V}, T$):
  $\forall \mathbf{V}_{h \to *}$: $C_{l \to n} = 1/N$
  for T iterations **do**
    $\forall \mathbf{\Omega}_n^{out}$: M-STEP($\mathbf{V}, C$) ▷ *hold C constant, adjust $(\boldsymbol{\mu}_n, \boldsymbol{\sigma}_n, A_n)$*
    $\forall \mathbf{V}_{h \to *}$: E-STEP($\mathbf{V}, \boldsymbol{\mu}, \boldsymbol{\sigma}, A$) ▷ *hold $(\boldsymbol{\mu}, \boldsymbol{\sigma}, A)$ constant, adjust $C_{h \to *}$*
    $\forall \mathbf{\Omega}_n^{out}$: $\mathbf{\Omega}_n^{out} = A_n * \boldsymbol{\mu}_n$
  return $\mathbf{\Omega}$



Sabour et al., *Dynamic Routing Between Capsules*, in NIPS 2017.
Hinton et al., *Matrix Capsules with EM Routing,* in ICLR 2017.

Two Complementary Work

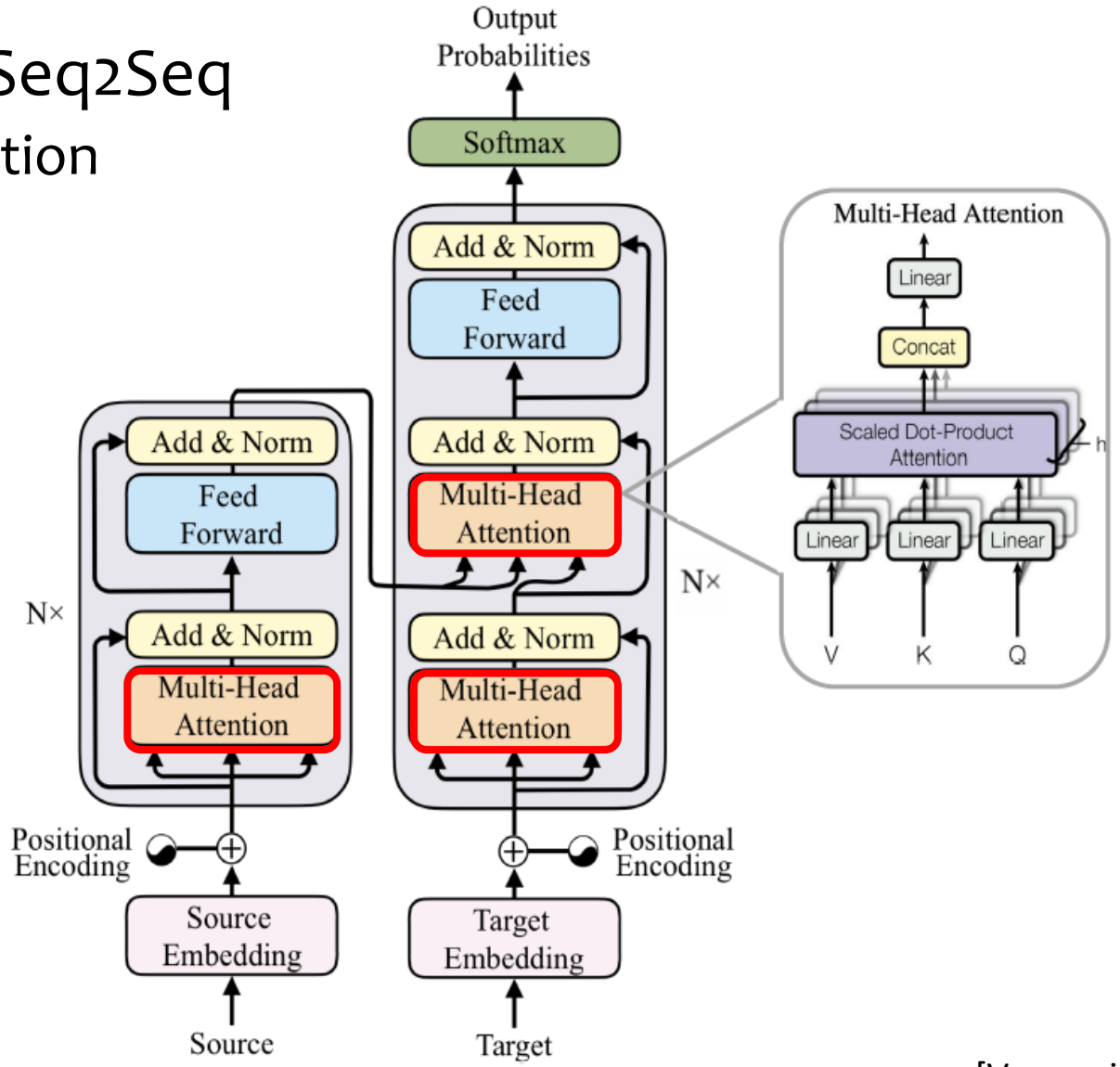
- Disagreement Regularization: [EMNLP'18]
  - improve information extraction
  - only adjust loss function

- Advanced Aggregation Function: [NAACL'19]
  - improve information aggregation
  - modify the network architecture

They are *complementary* to each other and can be applied simultaneously.

Experiments

- Transformer for Seq2Seq
  - Machine Translation



[Vaswani et al., *Attention is All You Need,* 2017]

Experiments

- Evaluation study on disagreement regularization:

#	Regularization			Speed	BLEU
	Sub.	*Pos.*	*Out.*		
1	×	×	×	1.21	24.13
2	✓	×	×	1.15	24.64
3	×	✓	×	1.14	24.42
4	×	×	✓	1.15	**24.78**
5	✓	×	✓	1.12	24.73
6	✓	✓	×	1.11	24.38
7	✓	✓	✓	1.05	24.60

Table 1: Effect of regularization terms, which are applied to the encoder self-attention only. "Speed" denotes the training speed (steps/second). Results are reported on the WMT17 Zh⇒En translation task using TRANSFORMER-BASE.

Only employing *output disagreement* is most effective (Row 4).

Experiments

- Evaluation study on advanced aggregation function:

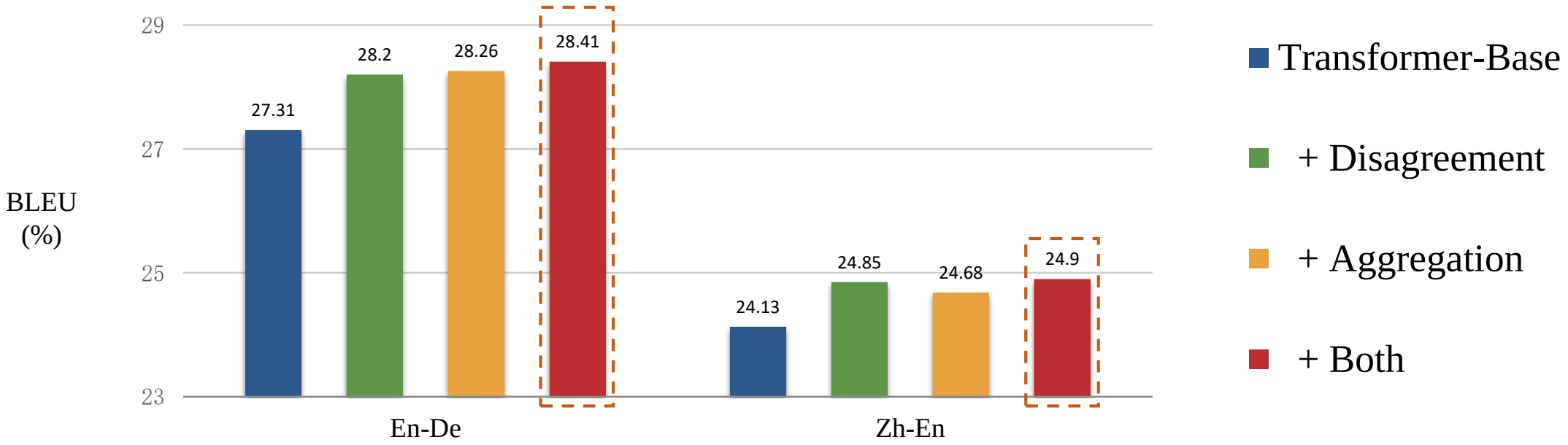
#	Applying Aggregation to ...			Routing	# Para.	Speed		BLEU	△
1	*Enc-Self*	*Enc-Dec*	*Dec-Self*			Train	Decode		
2	×	×	×	n/a	88.0M	1.92	1.67	27.31	–
3	✓	×	×	Simple	+12.6M	1.23	1.66	27.98	+0.67
4	✓	×	×	EM	+12.6M	1.20	1.65	28.28	+0.97
5	×	✓	×	EM	+12.6M	1.20	1.21	27.94	+0.63
6	×	×	✓	EM	+12.6M	1.21	1.21	28.15	+0.84
7	✓	✓	×	EM	+25.2M	0.87	1.20	28.45	+1.14
8	✓	✓	✓	EM	+37.8M	0.66	0.89	28.47	+1.16

Table 2: Effect of information aggregation on different attention components, i.e., encoder self-attention ("*Enc-Self*"), encoder-decoder attention ("*Enc-Dec*"), and decoder self-attention ("*Dec-Self*"). "# Para." denotes the number of parameters, and "Train" and "Decode" respectively denote the training speed (steps/second) and decoding speed (sentences/second).

Applying *EM Routing* at the encoder side best balances effectiveness and efficiency (Row 4).

Experiments

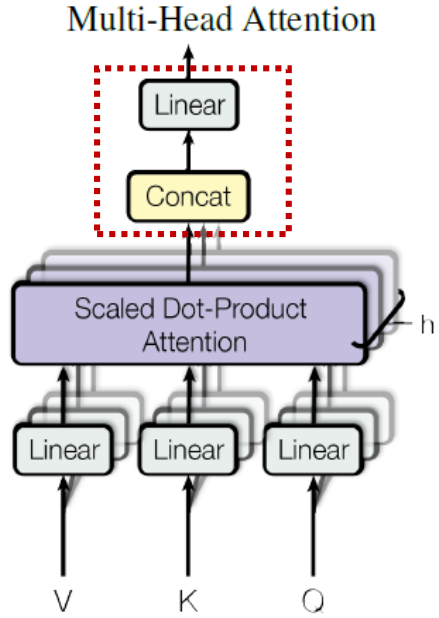
- Combining together and main results:



Summary

1. Propose disagreement regularization to improve the information extraction in multi-head attention.
2. Propose routing-by-agreement aggregation function to adjust the information aggregation in multi-head attention.
3. The two approaches are complementary to each other.

$$J(\theta) = \arg\max_{\theta} \{ \underbrace{L(\mathbf{y}|\mathbf{x};\theta)}_{likelihood} + \lambda * \underbrace{D(\mathbf{a}|\mathbf{x},\mathbf{y};\theta)}_{disagreement} \}$$



Outline

- Topic 1: Neural Attention for Code Completion
- Topic 2: Multi-Head Self-Attention
- Topic 3: Pre-trained Attention for Code Generation
- Conclusion and Future Work

Semantic Parsing

- Map natural language utterances to logical forms or executable code.
  - Natural language understanding

Player	Country	Points	Winnings($)
S. Stricker	United States	9000	1260000
K.J. Choi	South Korea	5400	756000
R. Sabbatini	South Africa	3400	4760000
M. Calca	United States	2067	289333
E. Els	South Africa	2067	289333

Question: What is the points of South Korea player?

SQL: `SELECT Points WHERE Country = South Korea`

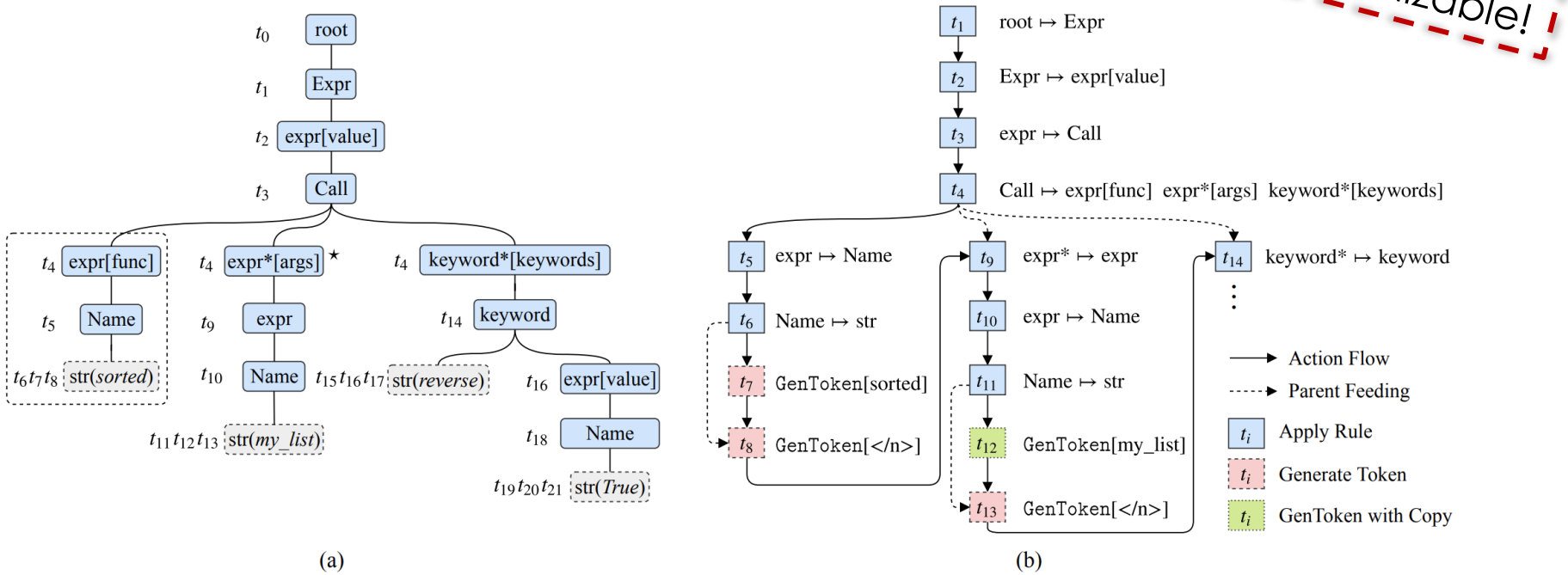
Answer: 5400

[Zhong et al., *Seq2SQL,* 2017]

Neural Semantic Parsing

- Neural Sequence-to-Sequence models:
  - Encoder: encode the natural language semantics
  - Decoder: generate the corresponding code
    - ➢ Sequential: bad performance due to lack of data
    - ➢ Syntax specific: external knowledge

Not Generalizable!



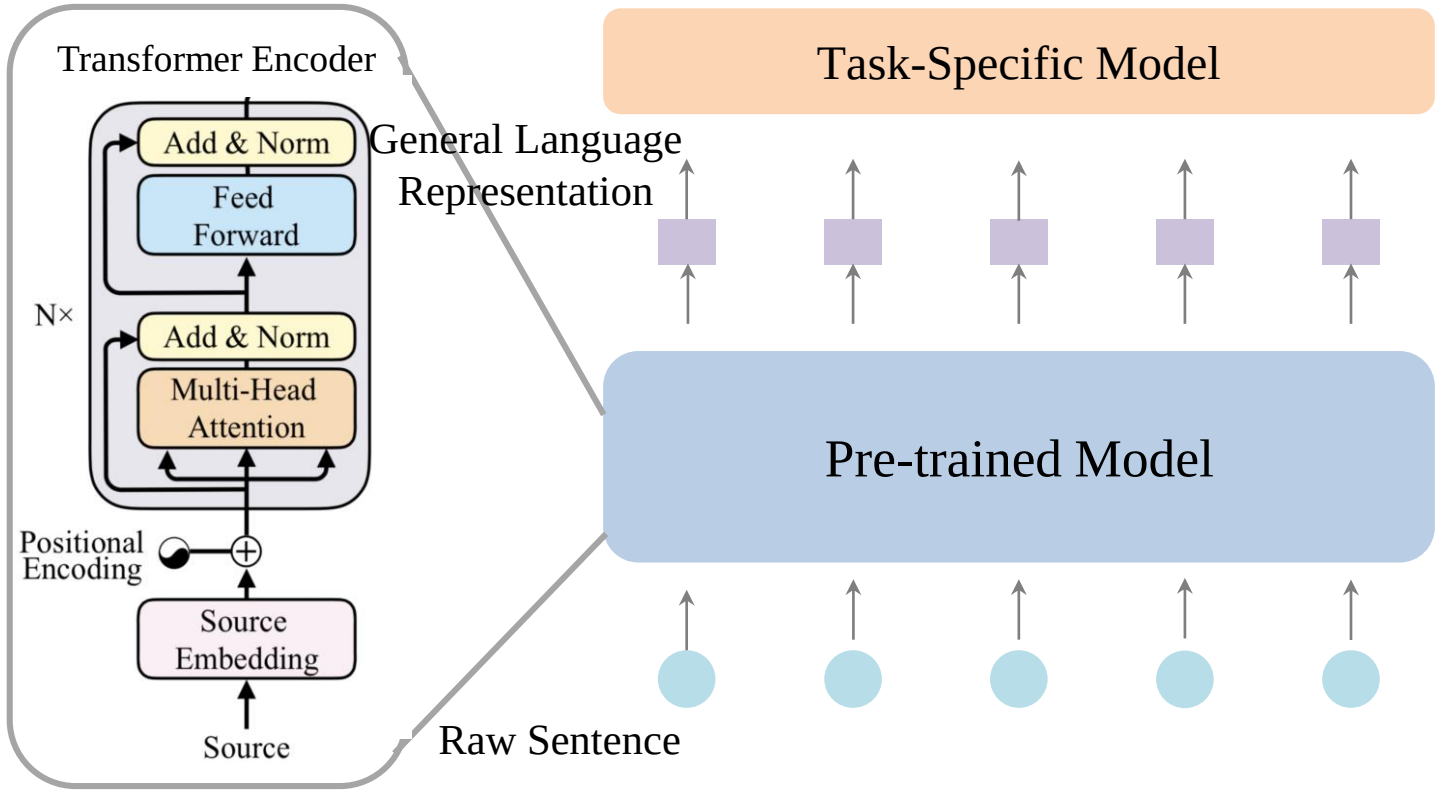
Input: `sort my_list in descending order`

Code: `sorted(my_list, reverse=True)`

[Yin et al., ACL 2017]

Pre-trained Models

- Pre-trained on large-scale text corpus.
  - Universal language representations
  - Self-attentional Transformer nets
  - BERT, GPT, XLNet, etc.



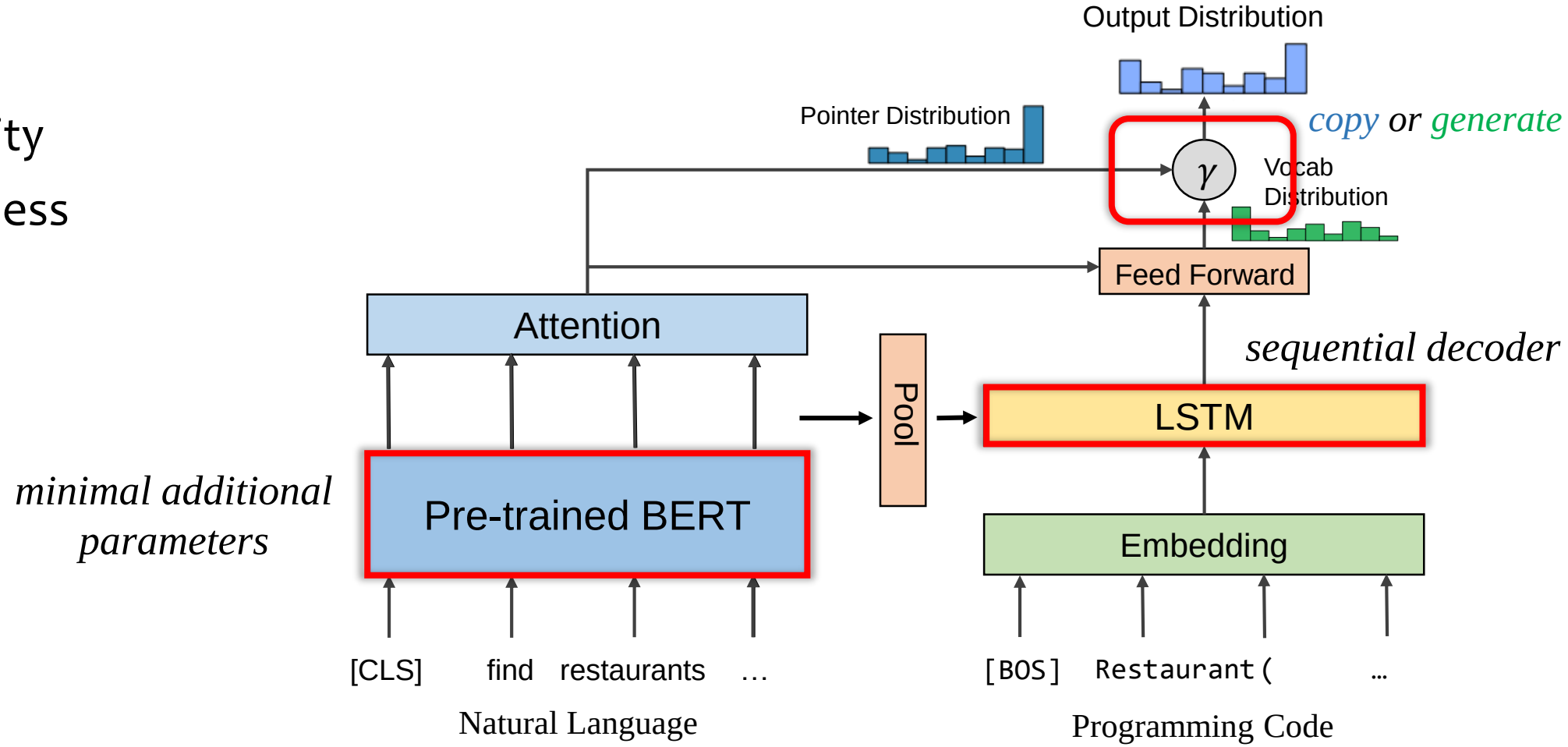
- Fine-tune on downstream tasks.
  - Transfer pre-trained knowledge
  - Limited data

Universal pre-trained attention models

Build semantic parsers that are both effective and generalizable?

Method

- BERT-LSTM
  - ❑Simplicity
  - ❑Extensibility
  - ❑Effectiveness



Experiments

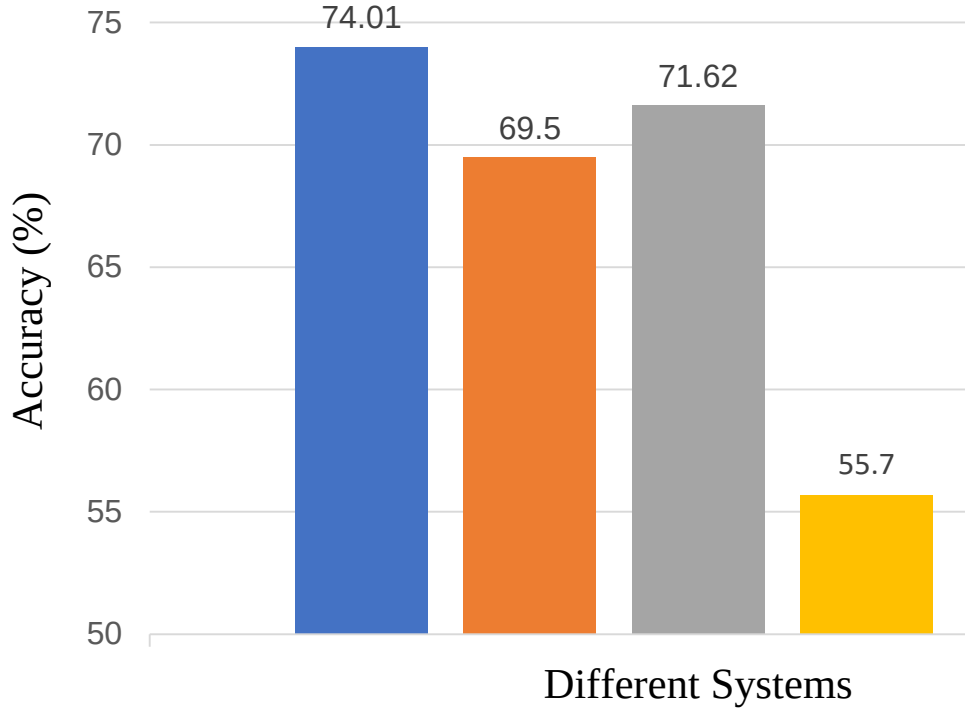
- Datasets:
  - Almond (Restaurant and People): NL question -> *ThingTalk* code
  - Django: NL description -> *Python* code
  - WiKiSQL: Table, NL Query -> *SQL* code

Input	*Table has columns: Conference Division Team City Home_Arena, which team is in the southeast with a home at Philips?*
Ref.	`SELECT (Team) FROM table WHERE Division = southeast AND Home_Arena = Philips`

- Metric: exact match accuracy

Evaluation Study

- LSTM decoder vs. Transformer decoder
- Greedy decoding vs. Beam search
- Fine-tune BERT vs. Freeze BERT



- Standard: LSTM+Greedy+Fine-tune
- - Change LSTM to Transformer
- - Change Greedy to Beam Search
- - Change Fine-tune to Freeze BERT

*Experiments on Almond-Restaurant

Experiments

- Accuracies on Almond and Django:

Model	ALMOND-RESTAURANT	ALMOND-PEOPLE
MQAN [102]	68.92%	75.65%
BERT-LSTM	**74.01%**	**81.93%**
– copying	56.76%	59.78%

Table 6.2: Code generation accuracies on the two ALMOND datasets.

Model	Accuracy
Sequence-to-Tree Network [34]	39.4%
Neural Machine Translation [162]	45.1%
Latent Predictor Network [94]	62.3%
Syntax Neural Model [162]	71.6%
Transition-Based Syntax Parser [163]	73.7%
Coarse-to-Fine Decoding [35]	74.1%
BERT-LSTM	**76.48%**
– copying	54.07%

Table 6.3: Python code generation accuracies on DJANGO. BERT-LSTM achieves state-of-the-art result.

Experiments

- Accuracies on WiKiSQL:

Model	Accuracy
Sequence-to-Sequence [171]	23.4%
Sequence-to-SQL [171]†	48.3%
SQLNet [154]†	61.3%
Transition-based Syntax Parser [163]†	68.6%
Coarse-to-Fine Decoding [35]†	71.7%
Multi-Task QA Network [102]	75.4%
SQLova [65]†*	83.6%
X-SQL [57]†*	86.0%
HydraNet [100]†*	**86.5%**
BERT-LSTM*	78.49%
– copying	35.99%

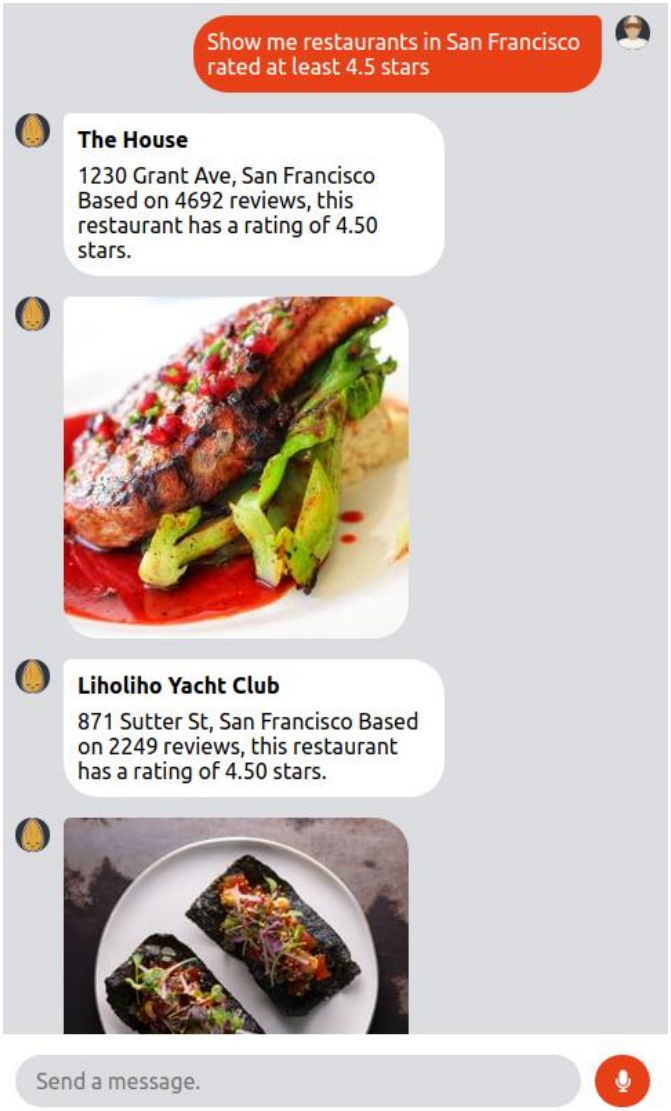
Table 6.4: SQL code generation accuracies on WikiSQL. "†" denotes syntax-specific models. "*" indicates that the model employs pre-trained BERT.

Case Study

- Almond Virtual Assistant

Input	*Show me restaurants in San Francisco rated at least 4.5 stars.*
Pred.	`now => (@org.schema.Restaurant.Restaurant) filter param:geo:Location == "San Francisco" and param:ratingValue:Number >= 4.5 => notify` ✔

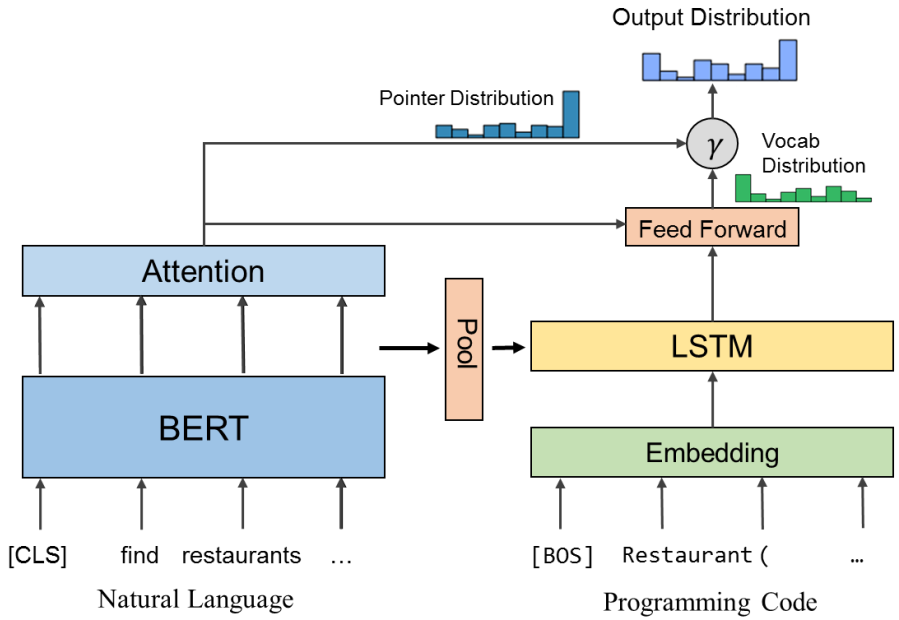
*Highlighted words: copying probability > 0.9



https://almond.stanford.edu/

Summary

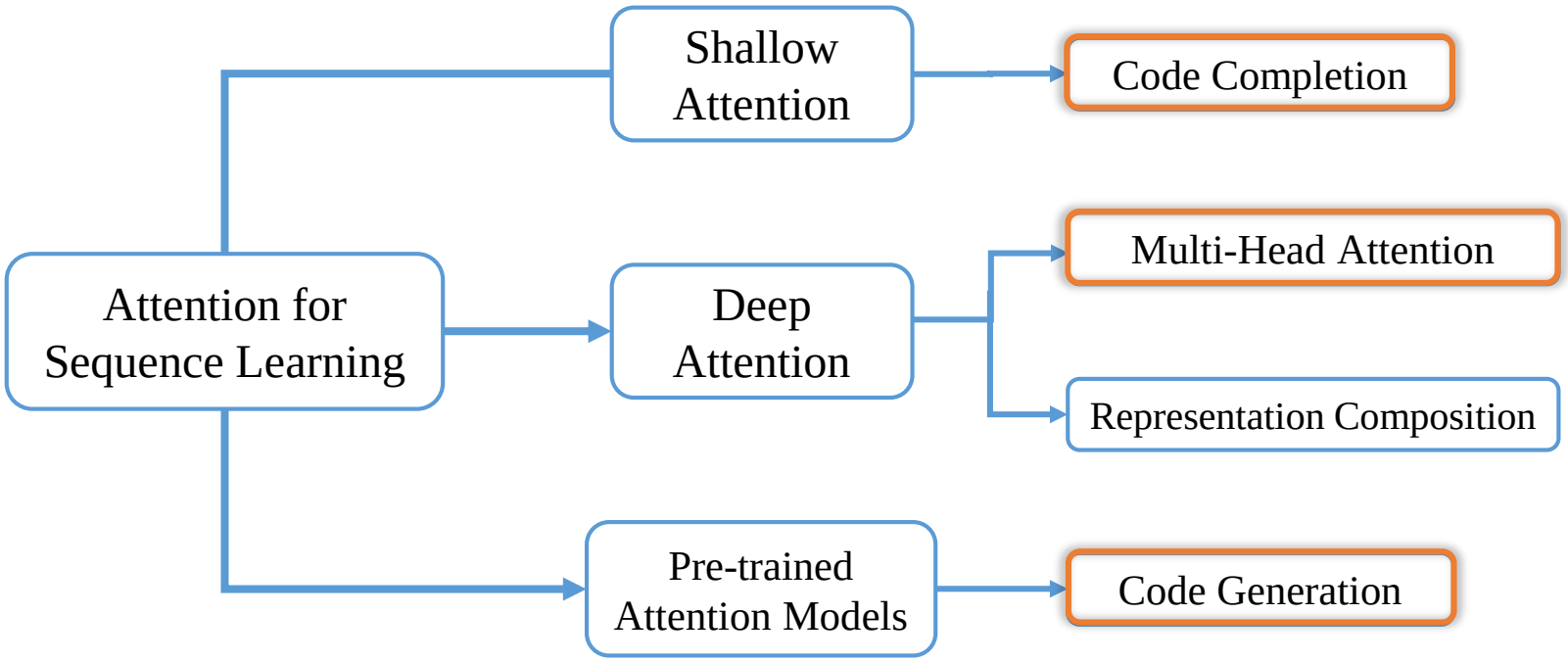
- Propose BERT-LSTM model for semantic parsing/code generation that is both effective and generalizable.
- Achieve state-of-the-art on three of the four experimental datasets.



Outline

- Topic 1: Neural Attention for Code Completion
- Topic 2: Multi-Head Self-Attention
- Topic 3: Pre-trained Attention for Code Generation
- Conclusion and Future Work

Conclusion



- Parent attention on AST
- Pointer mixture network
- Disagreement regularization
- Routing-by-agreement aggregation function
- BERT-LSTM model

Future Work

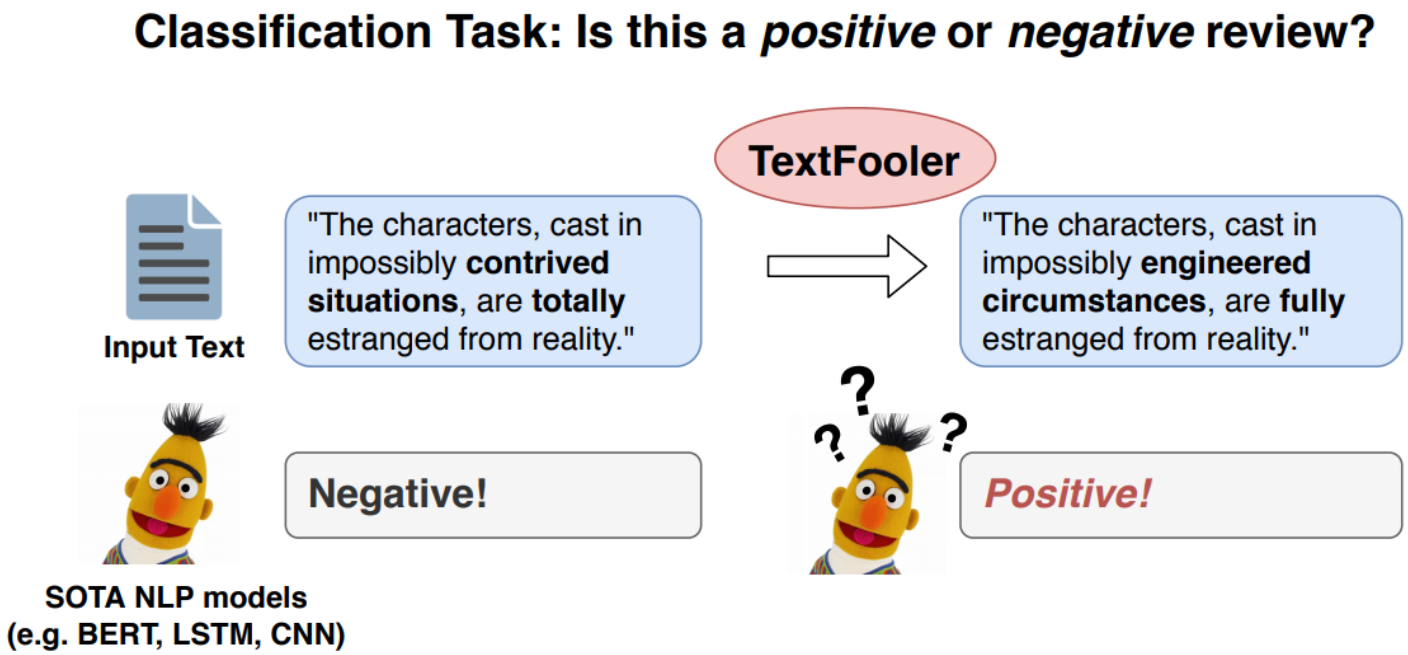
- Multi-Modal Attention Models
  - Textual and visual

A dog is standing on a hardwood floor.

A group of people sitting on a boat in the water.

Future Work

- Interpretability and Reliability of Attention Models
  - Adversarial attacks



Publications

1. **Jian Li**, Xing Wang, Baosong Yang, Shuming Shi, Michael R. Lyu, Zhaopeng Tu. *Neuron Interaction Based Representation Composition for Neural Machine Translation.* In AAAI 2020.
2. **Jian Li**, Baosong Yang, Zi-Yi Dou, Xing Wang, Michael R. Lyu, Zhaopeng Tu. *Information Aggregation for Multi-Head Attention with Routing-by-Agreement.* The 2019 North American Chapter of the Association for Computational Linguistics (NAACL), 2019.
3. Baosong Yang, **Jian Li**, Derek Wong, Lidia Chao, Xing Wang, Zhaopeng Tu. *Context-Aware Self-Attention Networks*. The 33rd AAAI Conference on Artificial Intelligence (AAAI), 2019.
4. **Jian Li**, Zhaopeng Tu, Baosong Yang, Michael R. Lyu and Tong Zhang. *Multi-Head Attention with Disagreement Regularization*. The 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2018.
5. **Jian Li**, Yue Wang, Michael R. Lyu, Irwin King. *Code Completion with Neural Attention and Pointer Networks.* The 28th International Joint Conference on Artificial Intelligence (IJCAI), 2018.
6. **Jian Li**, Pinjia He, Jieming Zhu, Michael R. Lyu. *Software Defect Prediction via Convolutional Neural Network.* The IEEE International Conference on Software Quality, Reliability and Security (QRS), 2017.
7. Pinjia He, Jieming Zhu, Shilin He, **Jian Li**, Michael R. Lyu. *Towards Automated Log Parsing for Large-Scale Log Data Analysis.* IEEE Transactions on Dependable and Secure Computing (TDSC), 2018.
8. Pinjia He, Jieming Zhu, Shilin He, **Jian Li**, Michael R. Lyu. *An Evaluation Study on Log Parsing and Its Use in Log Mining.* The 46th Annual International Conference on Dependable Systems and Networks (DSN), 2016.
9. Silei Xu, Giovanni Campagna, **Jian Li**, Monica S. Lam. *Schema2QA: Answering Complex Queries on the Structured Web with a Neural Model*. In submission to CIKM 2020.

物有本末，事有終始。知所先後，則近道矣。

Things have their roots and branches, affairs have their end and beginning. When you know what comes first and what comes last, then you are near the Way.

-《大學》

The Great Learning

Thanks!