Neural Responder, Answerer, Enquirer, Copier, etc.

Lili Mou
doublepower.mou@gmail.com

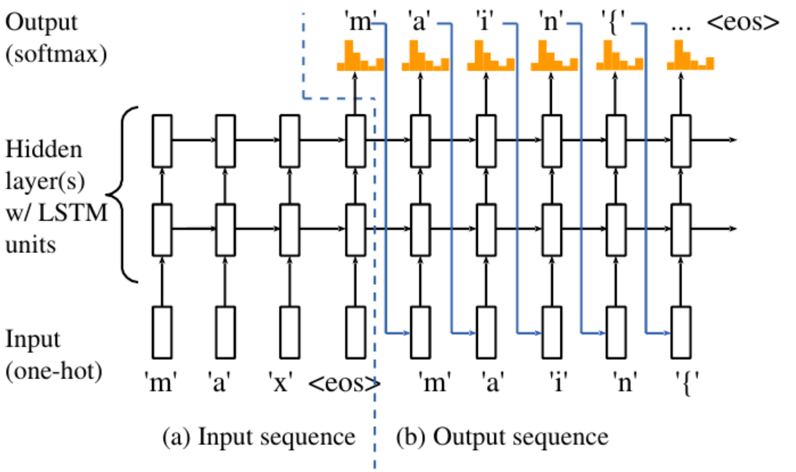
Outline

- Neural Responding Machine
- Neural Question Answering
- Neural Enquirer: Learning to Query Tables with Natural Language
- Incorporating Copying Mechanism in Sequence-to-Sequence Learning

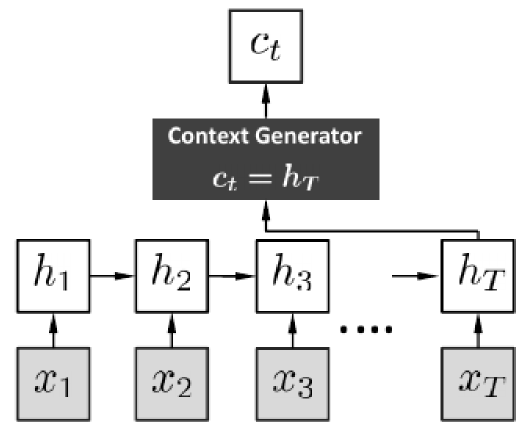
ACL'15

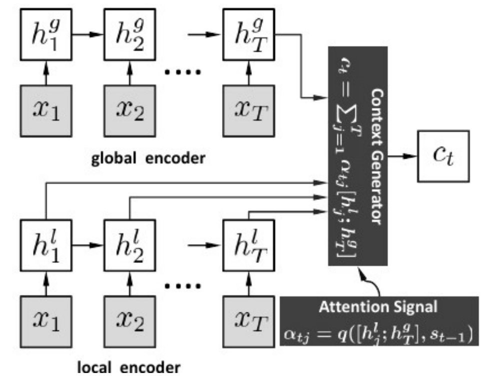
Neural Responder

- seq2seq



- Attention

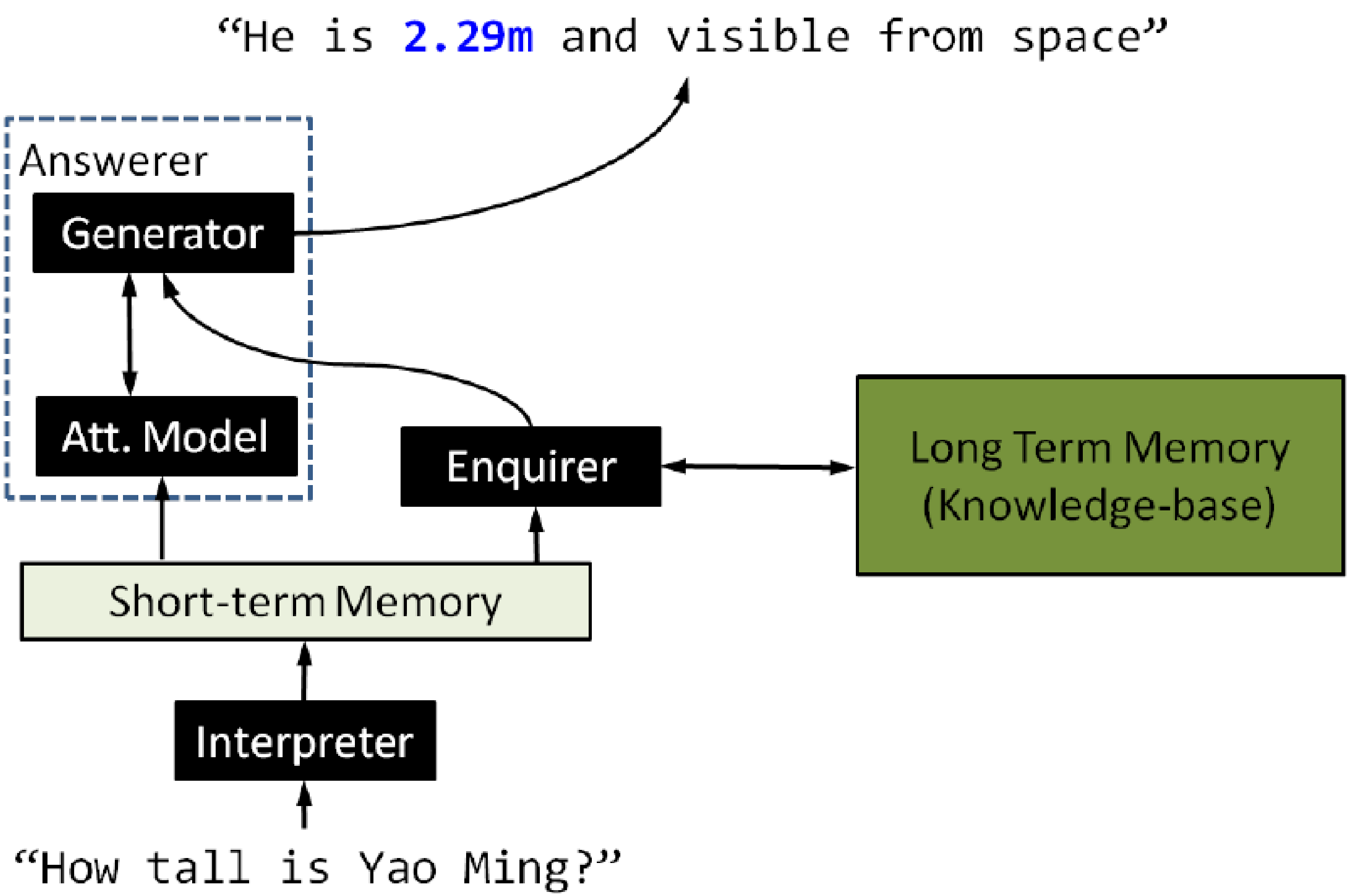




ID	Post	NRM-glo	NRM-loc	NRM-hyb	Rtr.-based
1	每年元旦都来高烧。 High fever attacks me every New Year's Day	多休息，增强免疫力！ Rest more to boost your immune system.	每年元旦都去医院看病 Go to see a doctor every New Year's Day	祝早日康复，身体健康！ Get well soon and stay healthy!	高烧… High fever…
2	又老了一岁，感谢我们组，我很高兴 I gain one more year. Grateful to my group, so happy	生日快乐！永远支持你！ Happy birthday! Will stand by you forever!	生日快乐！永远年轻漂亮！ Happy birthday! Wish you always stay young and pretty!	老了老了，岁月不饶人啊！ Getting old now. Time has no mercy.	我只是老了一岁 I just gain only one more year.

Neural Generative Question Answering

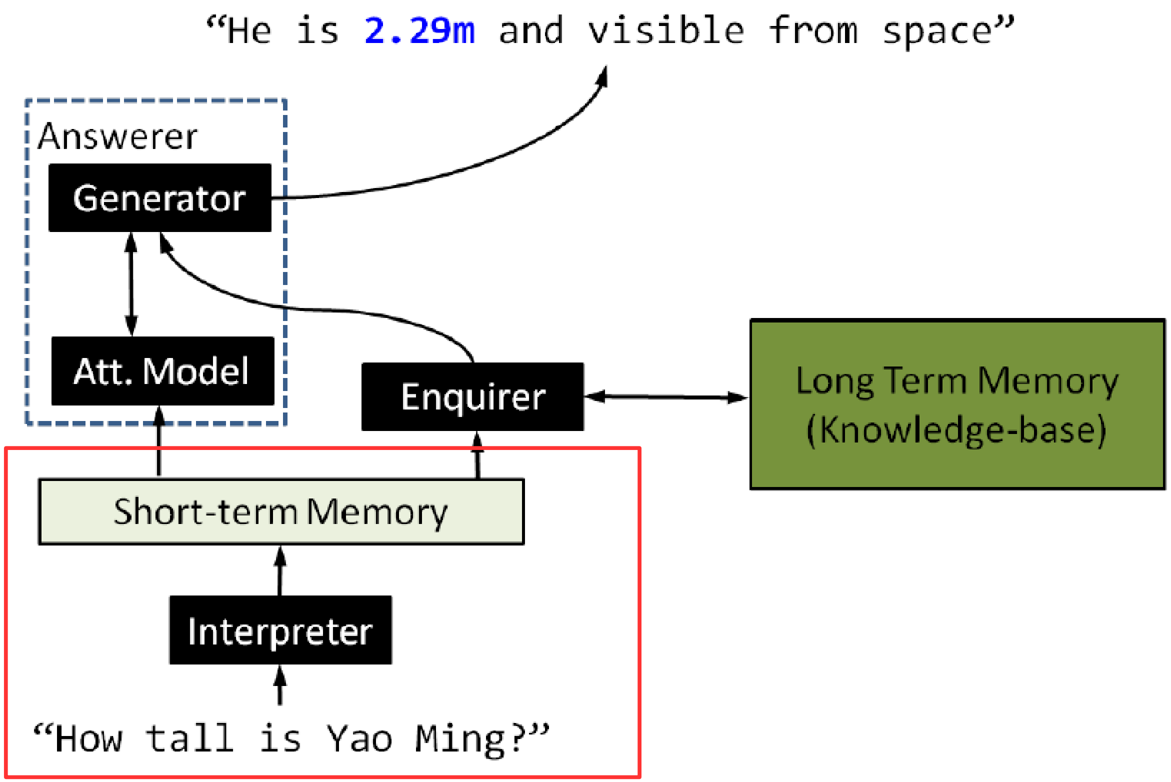
Jun Yin[1*] Xin Jiang[2] Zhengdong Lu[2]
Lifeng Shang[2] Hang Li[2] Xiaoming Li[1]
[1]School of Electronic Engineering and Computer Science, Peking University
{jun.yin, lxm}@pku.edu.cn
[2]Noah's Ark Lab, Huawei Technologies
{Jiang.Xin, Lu.Zhengdong, Shang.Lifeng, HangLi.HL}@huawei.com



Neural QA: Interpreter

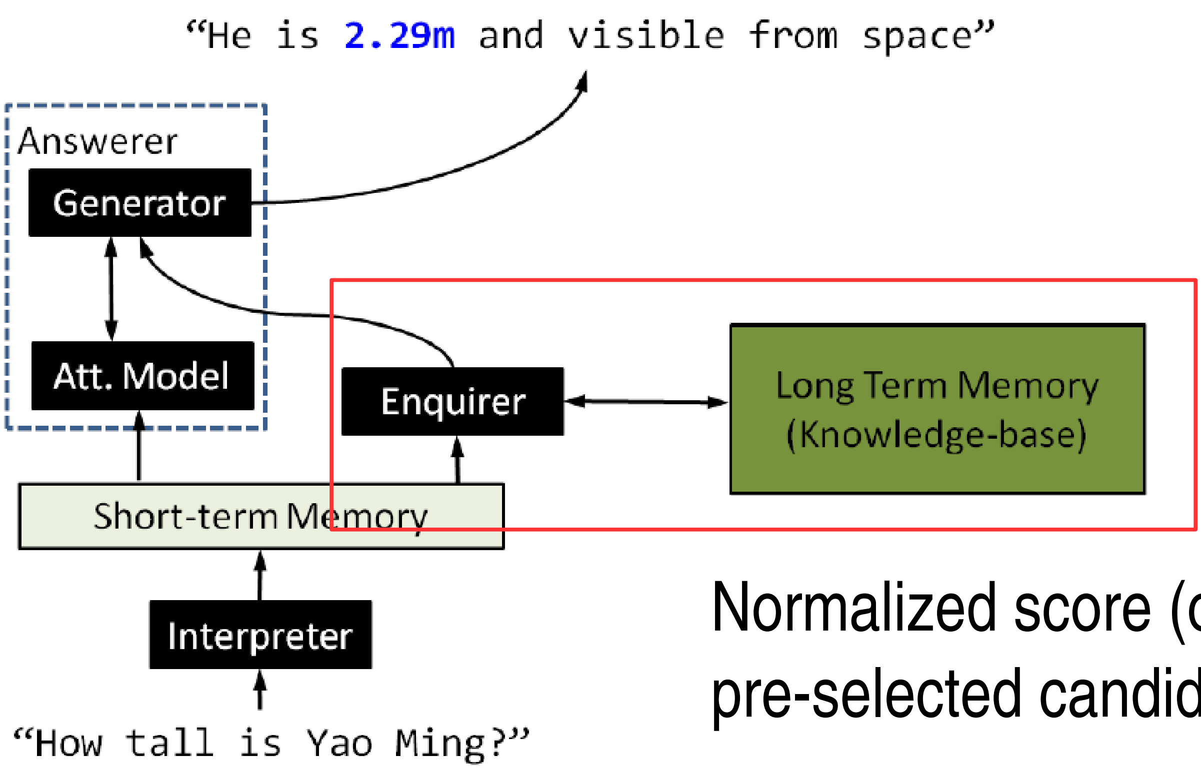
- Interpreter: Bi-RNN over the query

$$\tilde{\mathbf{h}}_t = [\mathbf{h}_t; \mathbf{x}_t]$$



Neural QA: Enquirer

- Interpreter: $\tilde{\mathbf{h}}_t = [\mathbf{h}_t; \mathbf{x}_t]$
- Enquirer: $S(Q, \tau) = \bar{\mathbf{h}}_Q^\top \mathbf{M} \mathbf{u}_\tau$



$\bar{\mathbf{h}}_Q$: Avg pooling of **h**

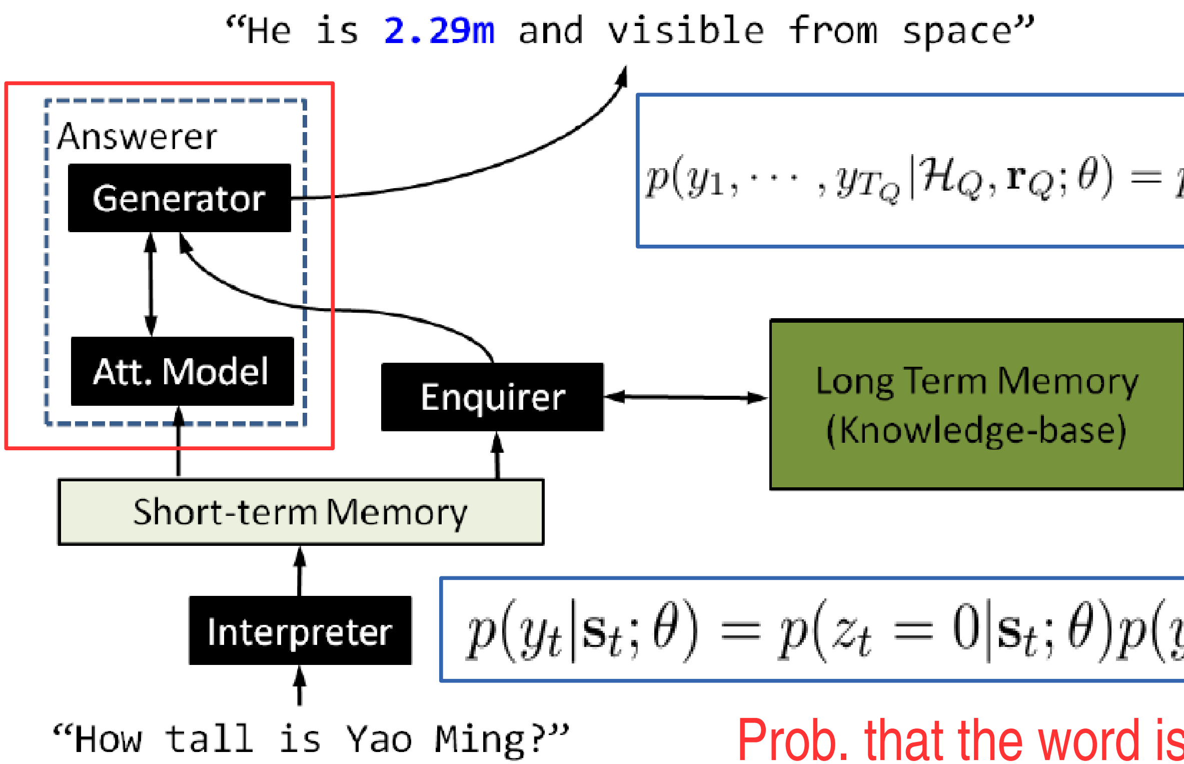
u: embedding of the tuple (sum of subject & predicate)

M: parameters

Normalized score (over 50 pre-selected candidates) $\dfrac{e^{S(Q,\tau_k)}}{\sum_{k'} e^{S(Q,\tau_{k'})}}$

Neural QA: Answerer

- Interpreter: $\tilde{\mathbf{h}}_t = [\mathbf{h}_t; \mathbf{x}_t]$
- Enquirer: $S(Q,\tau) = \bar{\mathbf{h}}_Q^\top \mathbf{M}\mathbf{u}_\tau$
- Answerer: RNN generator w/ attent.



$$p(y_1,\cdots,y_{T_Q}|\mathcal{H}_Q,\mathbf{r}_Q;\theta) = p(y_1|\mathcal{H}_Q,\mathbf{r}_Q;\theta)\prod_{t=2}^{T_Q} p(y_t|y_1,\dots,y_{t-1},\mathcal{H}_Q,\mathbf{r}_Q;\theta)$$

Prob. that the word is generated from KB $\dfrac{e^{S(Q,\tau_k)}}{\sum_{k'} e^{S(Q,\tau_{k'})}}$

$$p(y_t|\mathbf{s}_t;\theta) = \underbrace{p(z_t=0|\mathbf{s}_t;\theta)}_{\text{Prob. that the word is generated by RNN w/ attent.}} p(y_t|\mathbf{s}_t,0;\theta) + p(z_t=1|\mathbf{s}_t;\theta)p(y_t|\mathbf{s}_t,1;\theta)$$

Prob. that the word is generated by RNN w/ attent.

Table 3: Training and test accuracies

Models	Training	Test
Retrieval-based	40%	36%
NRM[5]	15%	19%
GENQA	46%	47%

Questions	Answers generated by GENQA
三国演义谁写的？ Who wrote the Romance of the Three Kingdoms?	当然是罗贯中写的啊。 Of course it is Luo Guanzhong who wrote it.
项羽是哪国人？ Which country does Xiang Yu belong to?	楚国的皇帝。 King of the Chu State.
还珠格格多少集？ How many episodes does My Fair Princess have?	一共24集。 24 episodes in total.

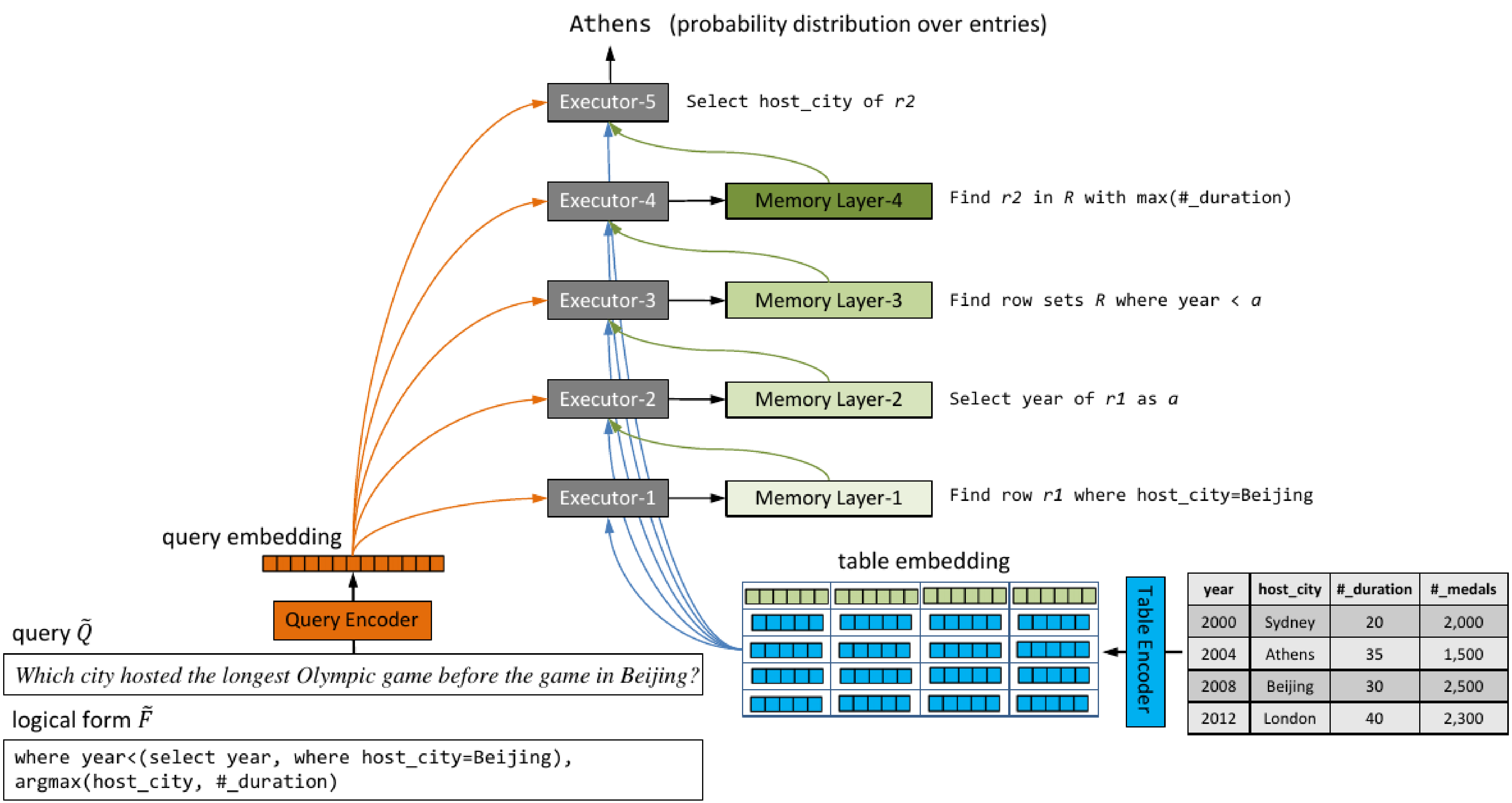
Outline

- Neural Responding Machine
- Neural Question Answering
- Neural Enquirer: Learning to Query Tables with Natural Language
- Incorporating Copying Mechanism in Sequence-to-Sequence Learning

NEURAL ENQUIRER: Learning to Query Tables with Natural Language

Disclaimer: Details may vary.

Pengcheng Yin[†*] Zhengdong Lu[‡] Hang Li[‡] Ben Kao[†]
[†]Dept. of Computer Science
The University of Hong Kong
{pcyin, kao}@cs.hku.hk
[‡]Noah's Ark Lab, Huawei Technologies
{Lu.Zhengdong, HangLi.HL}@huawei.com



Query Encoder

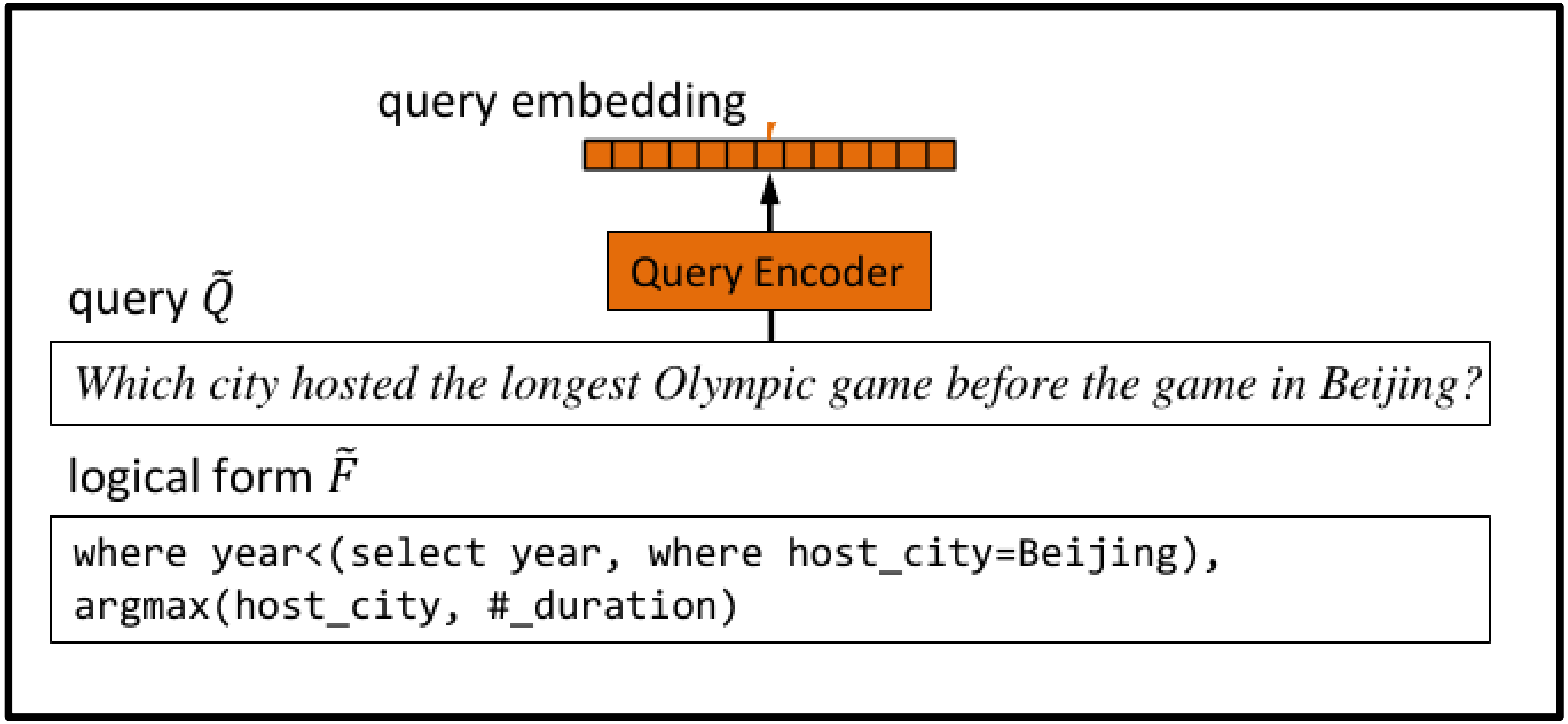
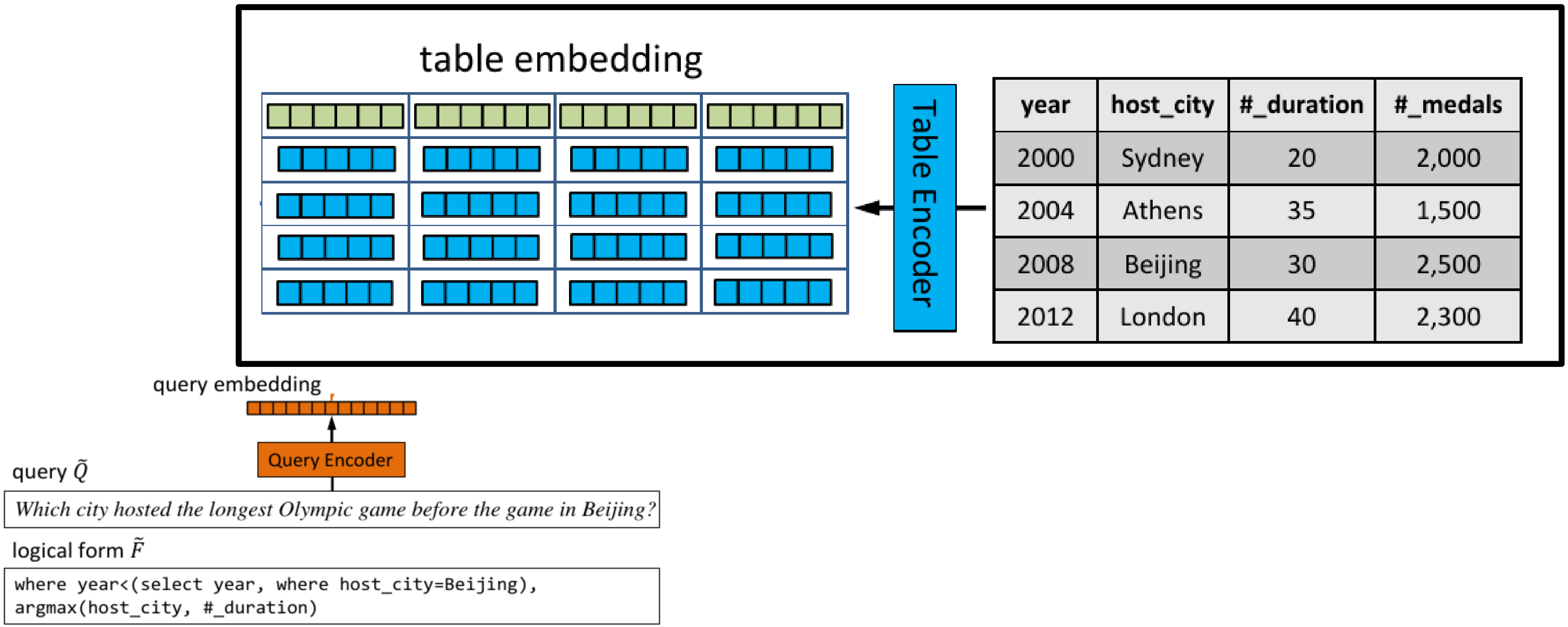
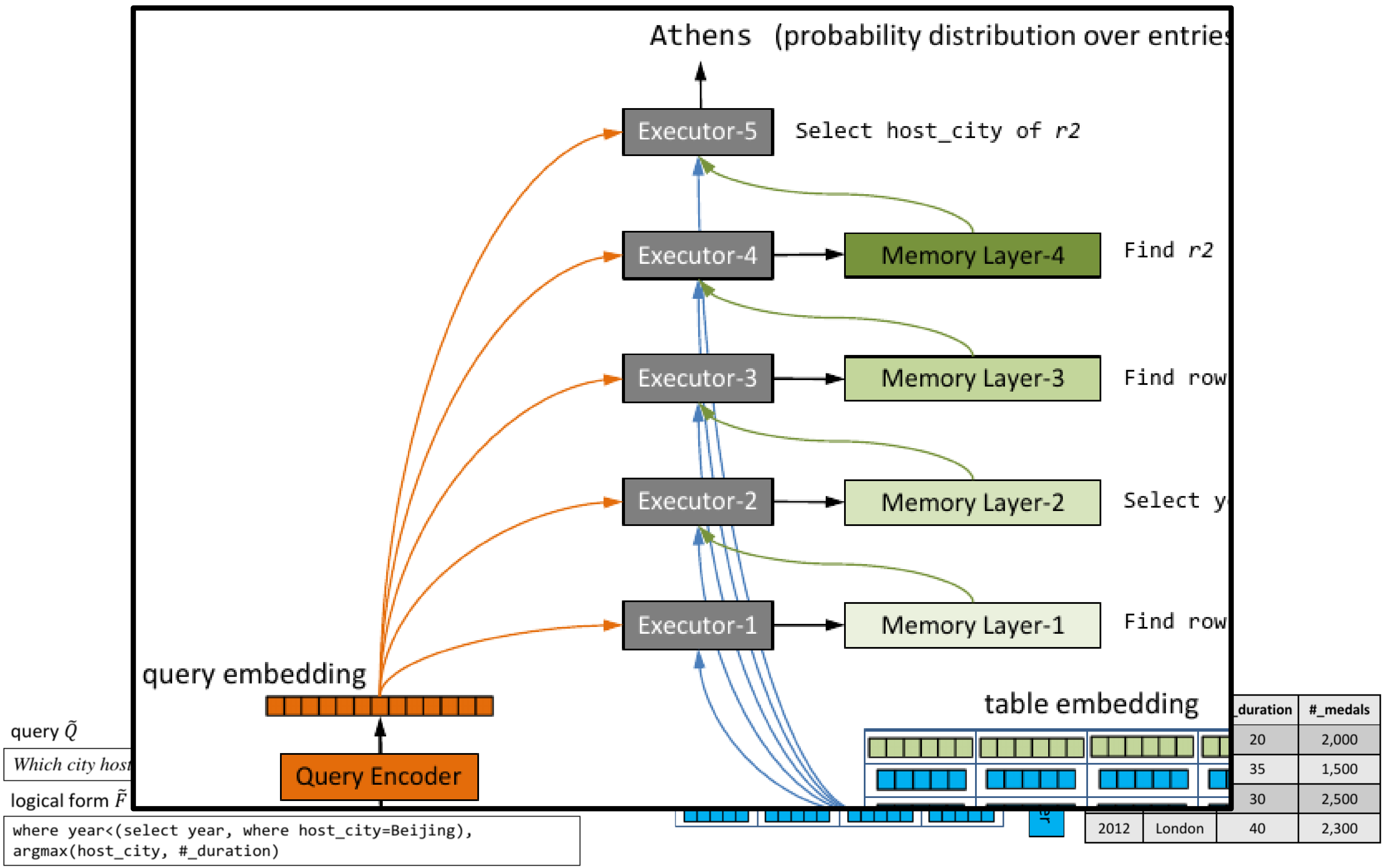


Table Encoder



Executor


Athens (probability distribution over entries
Executor-5
Select host_city of r2
Executor-4
Memory Layer-4
Find r2
Executor-3
Memory Layer-3
Find row
Executor-2
Memory Layer-2
Select y
Executor-1
Memory Layer-1
Find row
query embedding
table embedding
Query Encoder
query $\tilde{Q}$
Which city host
logical form $\tilde{F}$
where year<(select year, where host_city=Beijing),
argmax(host_city, #_duration)
_duration
#_medals
20
2,000
35
1,500
30
2,500
2012
London
40
2,300

Query Encoder

- BiLSTM for query (Q) parsing
- Logical form (F) appears to be used for supervision only

$$\mathbf{h}_t = \mathbf{z}_t \mathbf{h}_{t-1} + (\mathbf{1} - \mathbf{z}_t)\tilde{\mathbf{h}}_t$$
$$\tilde{\mathbf{h}}_t = \tanh(\mathbf{W}\mathbf{x}_t + \mathbf{U}(\mathbf{r}_t \circ \mathbf{h}_{t-1}))$$
$$\mathbf{z}_t = \sigma(\mathbf{W}_z \mathbf{x}_t + \mathbf{U}_z \mathbf{h}_{t-1})$$
$$\mathbf{r}_t = \sigma(\mathbf{W}_r \mathbf{x}_t + \mathbf{U}_r \mathbf{h}_{t-1})$$

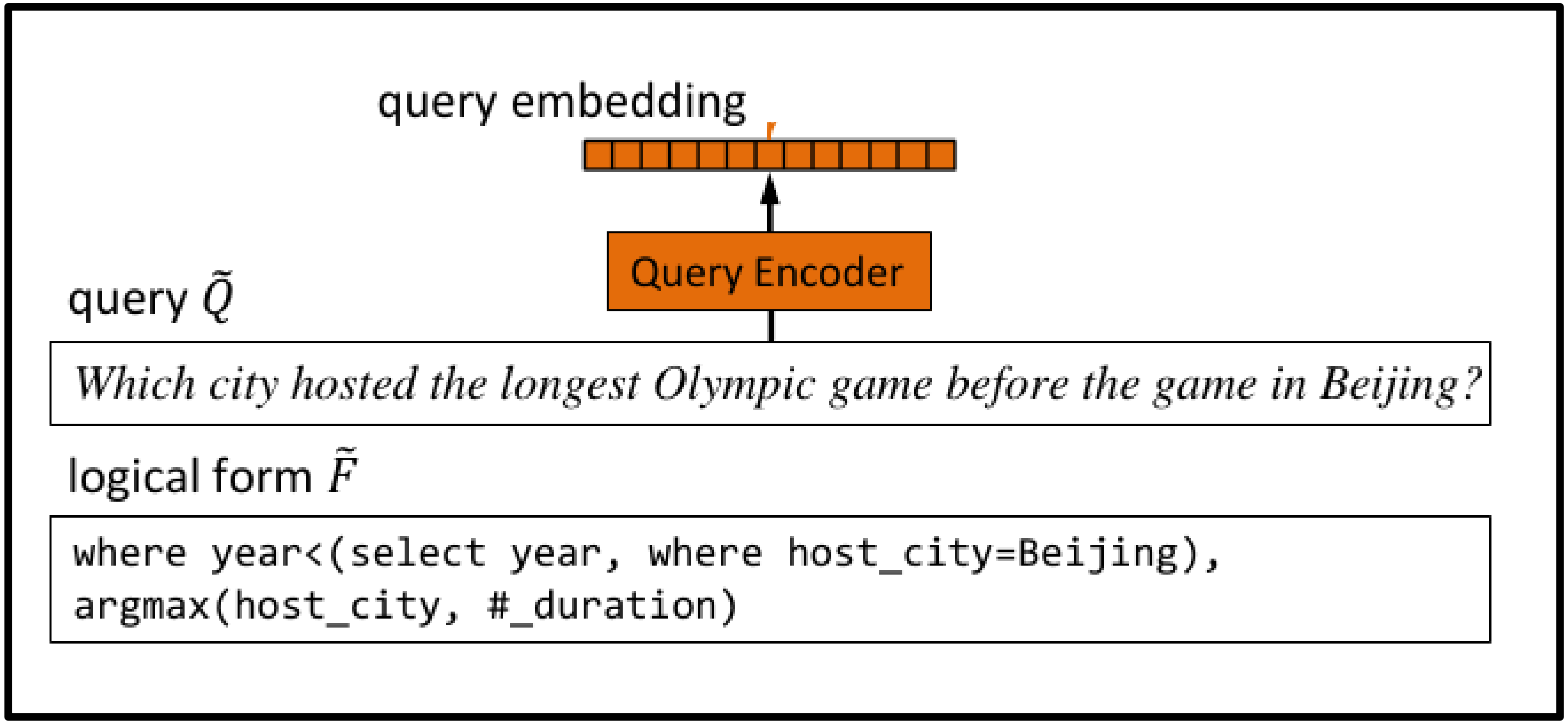
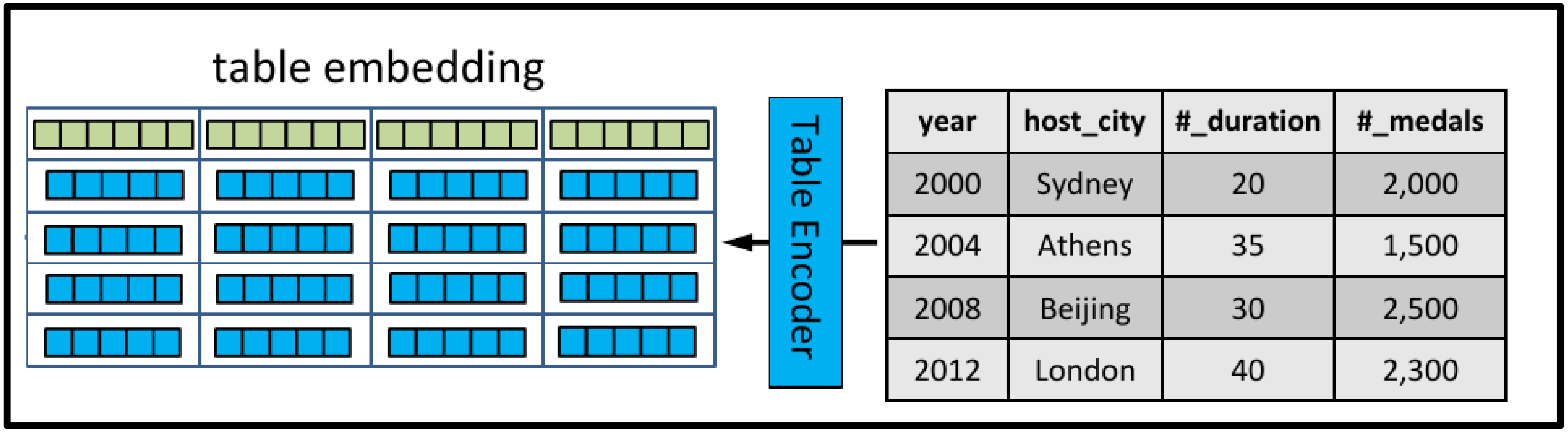


Table Encoder

- Vector representation of each entry

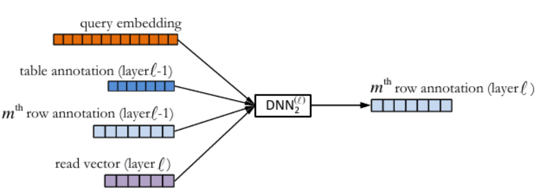
  (a function of the entry and the field)

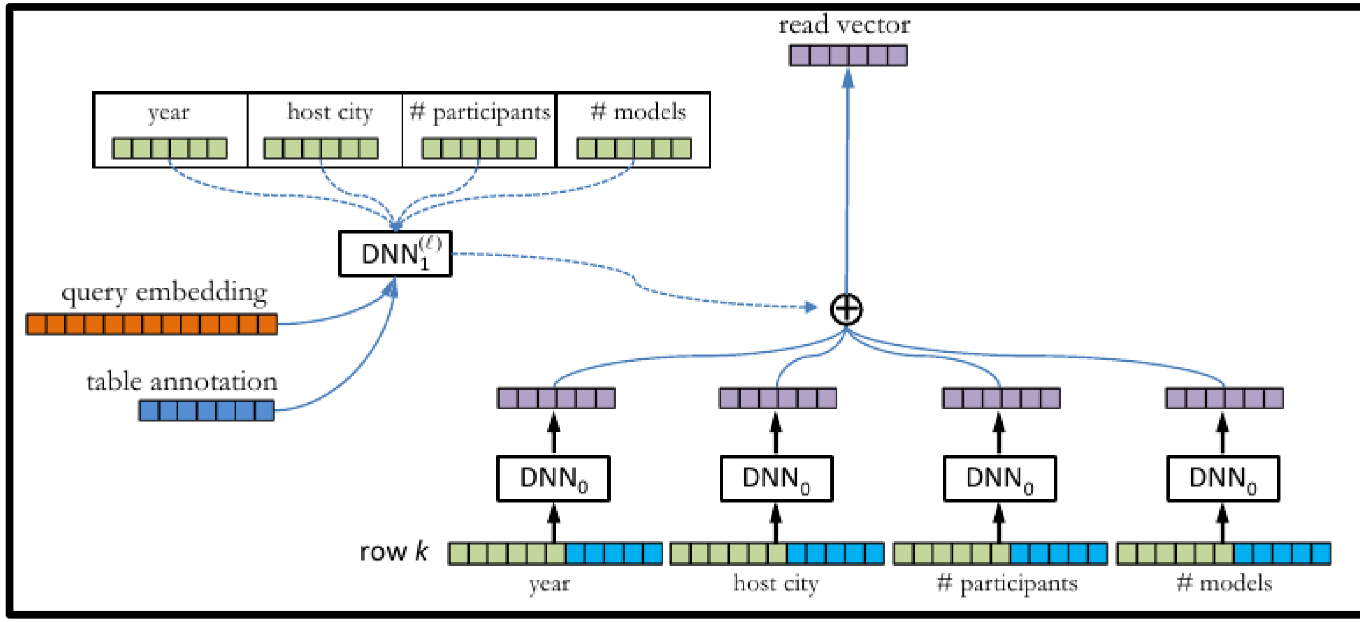
$$\mathbf{e}_{mn} = \mathrm{DNN}_0([\mathbf{L}[w_{mn}]; \mathbf{f}_n]) = \tanh(\mathbf{W} \cdot [\mathbf{L}[w_{mn}]; \mathbf{f}_n] + \mathbf{b})$$

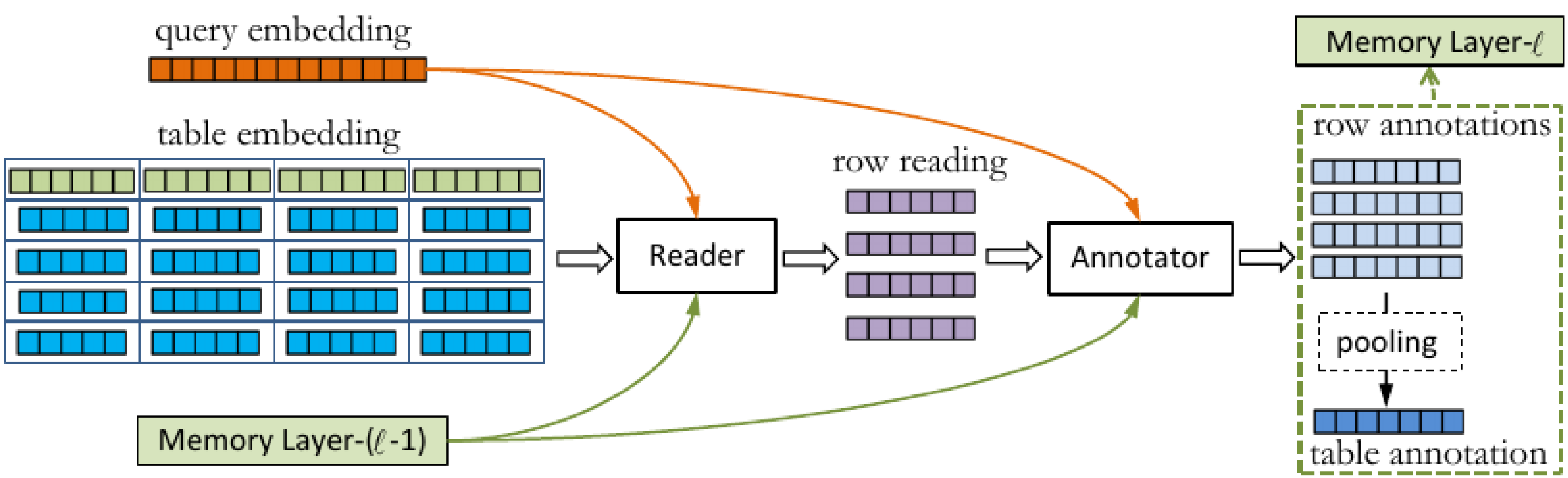


Executor

- **Reader**: To obtain a vector representation of a row
- **Annotator**:





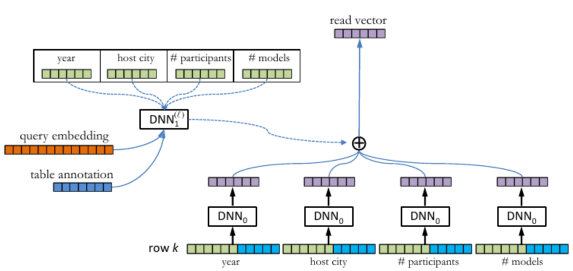


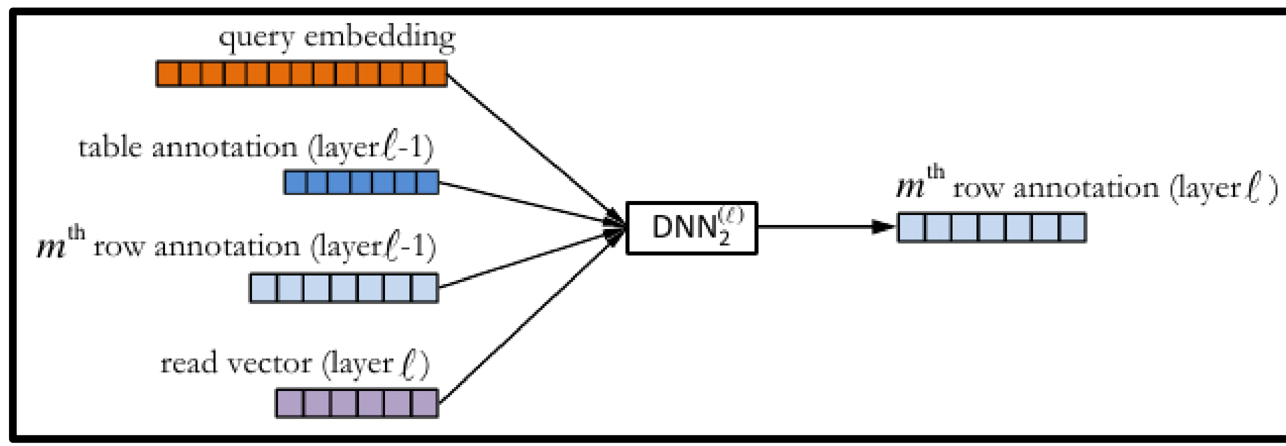
Executor

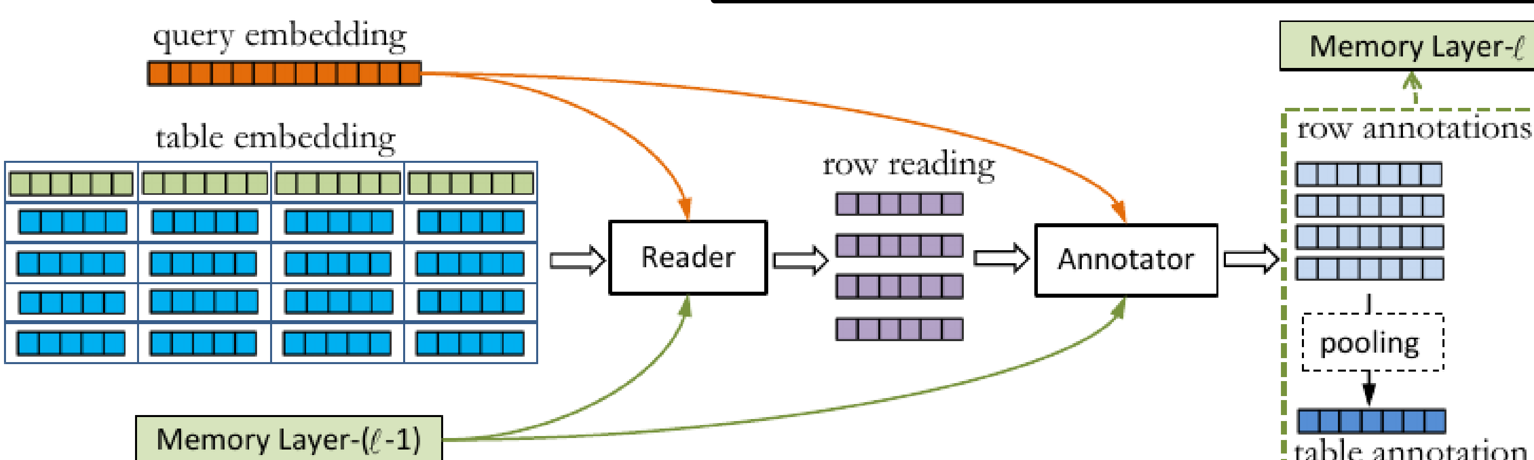
- **Reader**: To obtain a vector representation of a row
- **Annotator**: To obtain yet another vector representation

Read Vector: $\mathbf{r}_m^\ell = f_{\mathrm{R}}^\ell(\mathcal{R}_m, \mathcal{F}_\mathcal{T}, \mathbf{q}, \mathcal{M}^{\ell-1})$

Row Annotation: $\mathbf{a}_m^\ell = f_{\mathrm{A}}^\ell(\mathbf{r}_m^\ell, \mathbf{q}, \mathcal{M}^{\ell-1})$







Executor->Reader

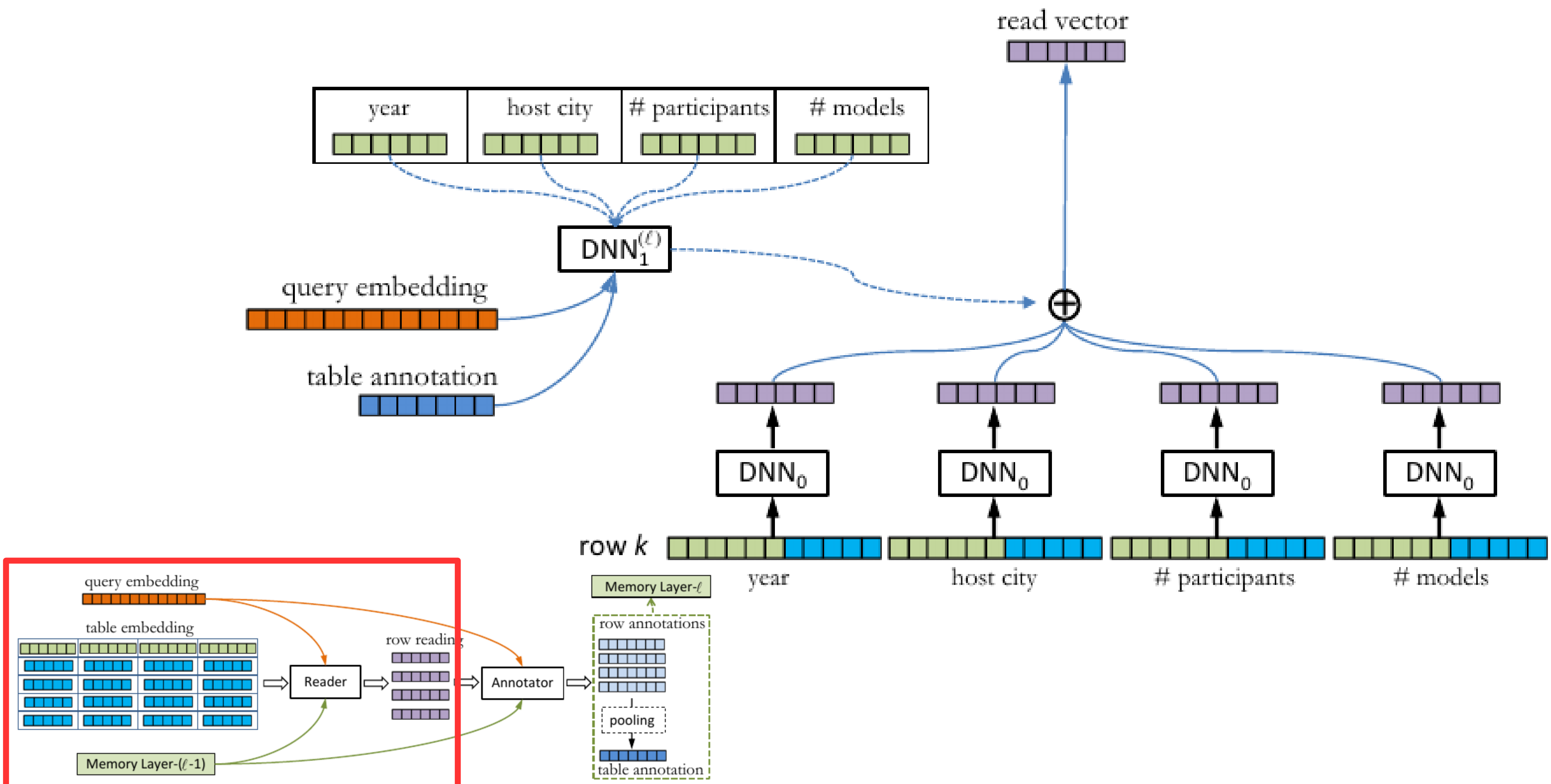
Read Vector: $\mathbf{r}_m^\ell = f_\mathrm{R}^\ell(\mathcal{R}_m, \mathcal{F}_\mathcal{T}, \mathbf{q}, \mathcal{M}^{\ell-1})$

- Select a column in each entry

$$\mathbf{r}_m^\ell = f_\mathrm{R}^\ell(\mathcal{R}_m, \mathcal{F}_\mathcal{T}, \mathbf{q}, \mathcal{M}^{\ell-1}) = \sum_{n=1}^{N} \tilde{\omega}(\mathbf{f}_n, \mathbf{q}, \mathbf{g}^{\ell-1})\mathbf{e}_{mn}$$

$$\tilde{\omega}(\mathbf{f}_n, \mathbf{q}, \mathbf{g}^{\ell-1}) = \frac{\exp(\omega(\mathbf{f}_n, \mathbf{q}, \mathbf{g}^{\ell-1}))}{\sum_{n'=1}^{N} \exp(\omega(\mathbf{f}_{n'}, \mathbf{q}, \mathbf{g}^{\ell-1}))}$$

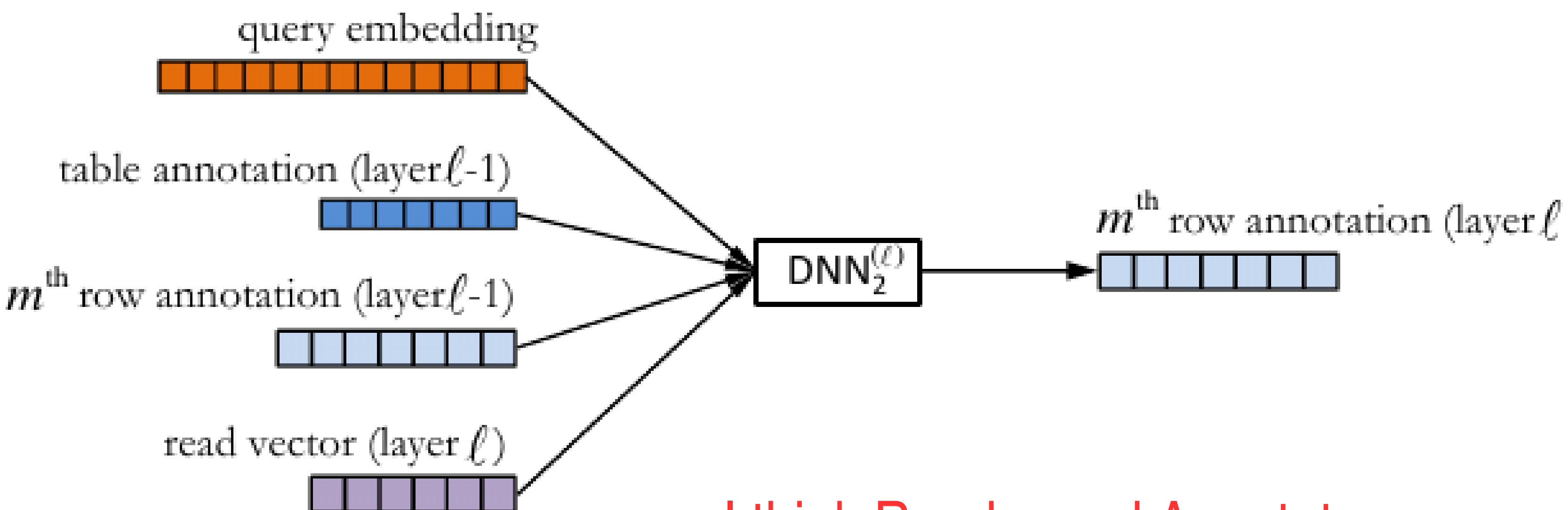
$\omega(\cdot)$ is modeled as a DNN (denoted as $\mathrm{DNN}_1^{(\ell)}$)



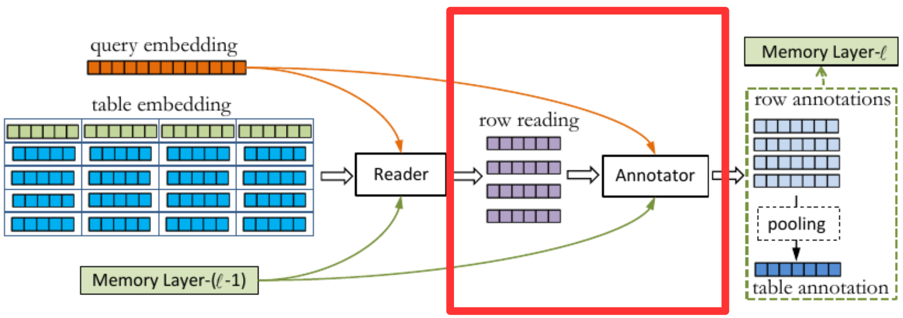
Executor->Row Annotator

$$\mathbf{a}_m^{\ell} = f_{\mathrm{A}}^{\ell}(\mathbf{r}_m^{\ell}, \mathbf{q}, \mathcal{M}^{\ell-1}) = \mathsf{DNN}_2^{(\ell)}([\mathbf{r}_m^{\ell}; \mathbf{q}; \mathbf{a}_m^{\ell-1}; \mathbf{g}^{\ell-1}])$$

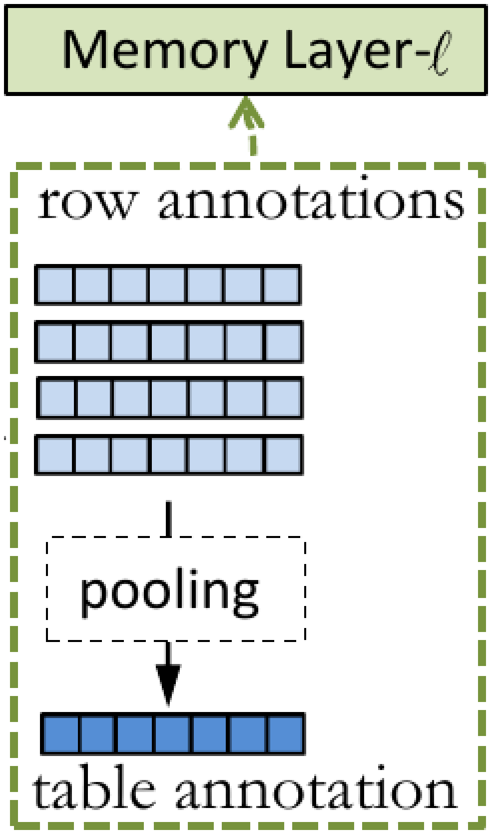
- More complex information mix than Reader



I think Reader and Annotator are compensatory to some extent, e.g., a 2-D attention mechanism (see Latent Predictor Network)

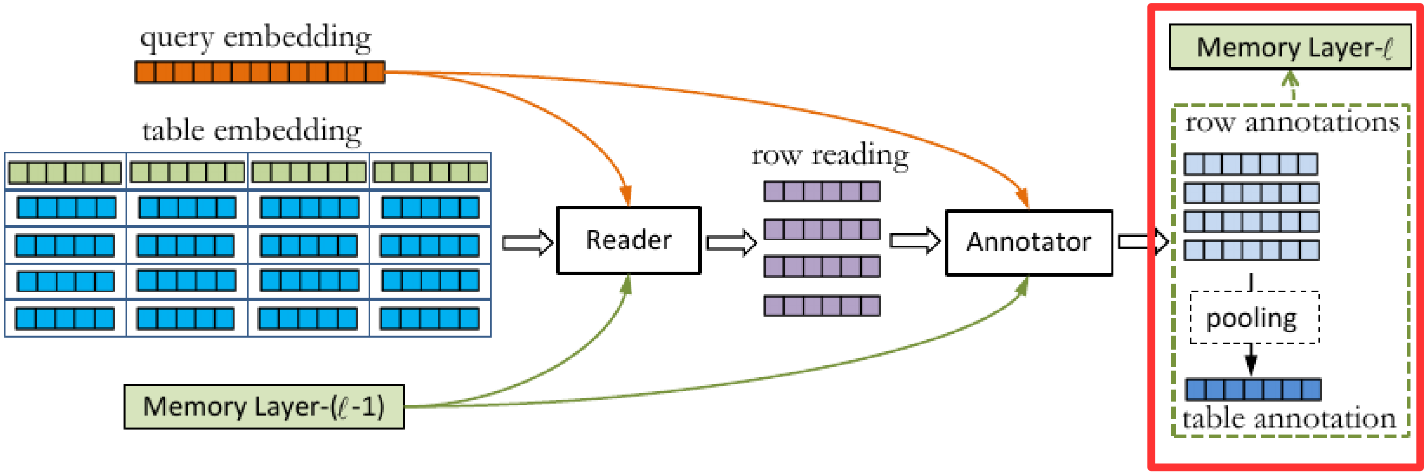


Executor->Table Annotator



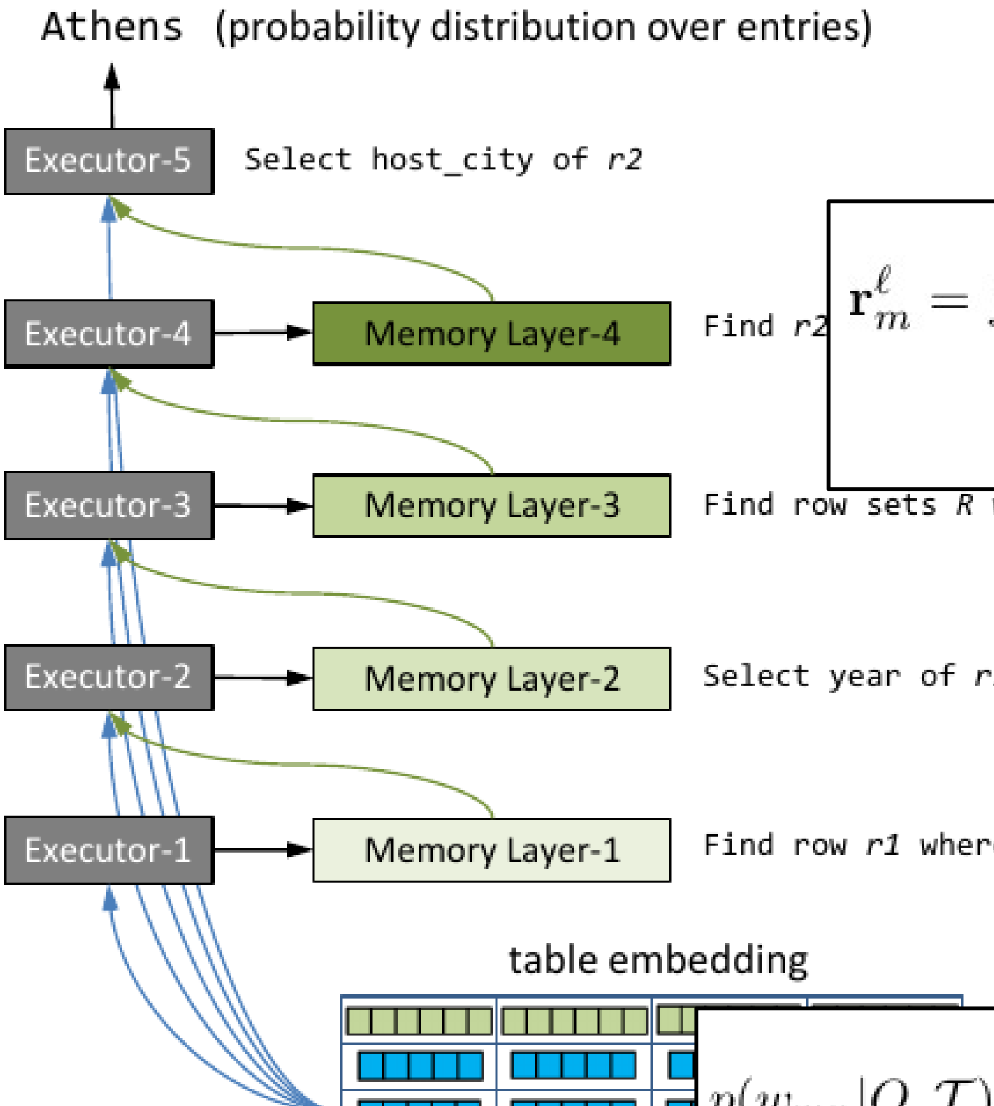
- Pooling vector

$$\mathbf{g}^{\ell} = f_{\mathrm{POOL}}(\mathbf{a}_1^{\ell}, \mathbf{a}_2^{\ell}, \ldots, \mathbf{a}_M^{\ell}) = [g_1, g_2, \ldots, g_{d_{\mathcal{G}}}]^{\top}$$



How is the table annotation used?

5 executors (predefined)



- For intermediate layers (1--4), g is stored in memory, and used when computing the next layer's Reader

$$\mathbf{r}_m^\ell = f_{\mathrm{R}}^\ell(\mathcal{R}_m, \mathcal{F}_\mathcal{T}, \mathbf{q}, \mathcal{M}^{\ell-1}) = \sum_{n=1}^{N} \tilde{\omega}(\mathbf{f}_n, \mathbf{q}, \boxed{\mathbf{g}^{\ell-1}})\mathbf{e}_{mn}$$

$$\tilde{\omega}(\mathbf{f}_n, \mathbf{q}, \mathbf{g}^{\ell-1}) = \frac{\exp(\omega(\mathbf{f}_n, \mathbf{q}, \boxed{\mathbf{g}^{\ell-1}}))}{\sum_{n'=1}^{N} \exp(\omega(\mathbf{f}_{n'}, \mathbf{q}, \boxed{\mathbf{g}^{\ell-1}}))}$$

- For the last layer, g is used to compute a probabilistic distribution over the entire table.

$$p(w_{mn}|Q, \mathcal{T}) = \frac{\exp(f_{\mathrm{ANS}}^\ell(\mathbf{e}_{mn}, \mathbf{q}, \mathbf{a}_m^{\ell-1}, \mathbf{g}^{\ell-1}))}{\sum_{m'=1}^{M} \sum_{n'=1}^{N} \exp(f_{\mathrm{ANS}}^\ell(\mathbf{e}_{m'n'}, \mathbf{q}, \mathbf{a}_{m'}^{\ell-1}, \mathbf{g}^{\ell-1}))}$$

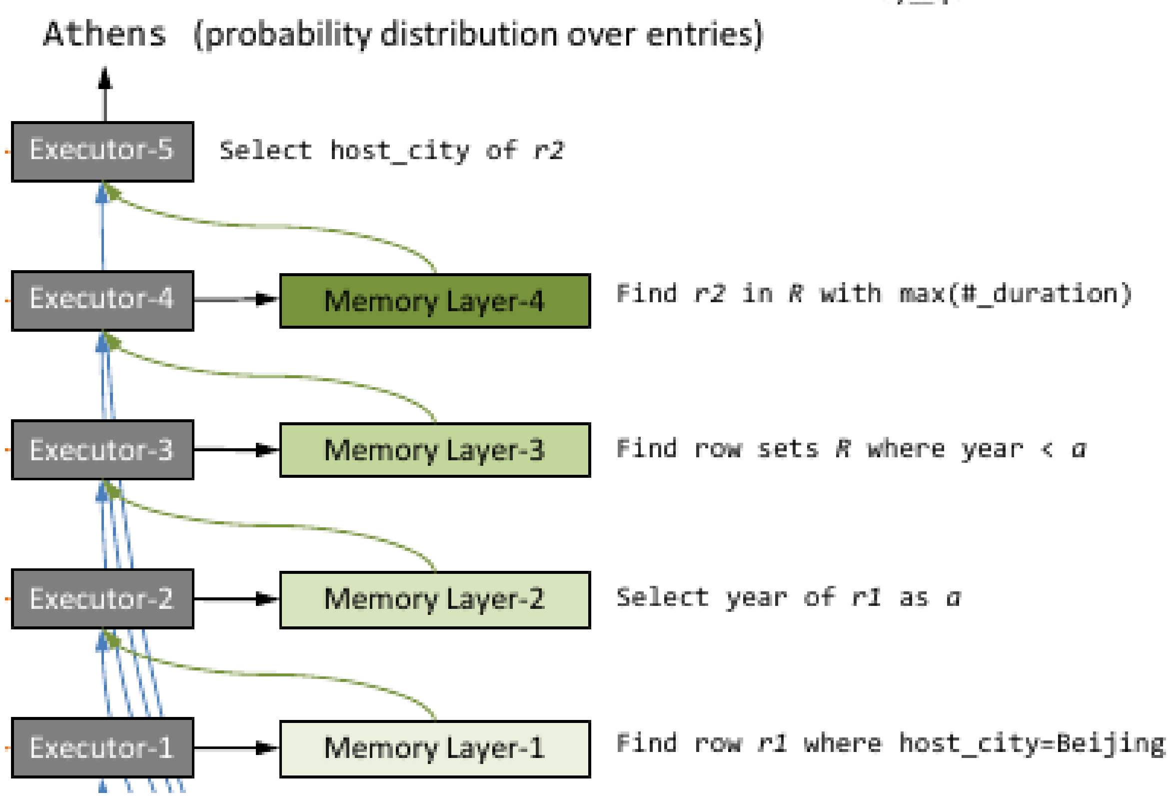
Training Objective

- End-to-end learning (N2N)

$$\mathcal{L}_{\mathrm{N2N}}(\mathcal{D}) = \sum_{i=1}^{N_{\mathcal{D}}} \log p(y^{(i)} = w_{mn} | Q^{(i)}, \mathcal{T}^{(i)})$$

- Step-by-step learning (SbS)

$$\mathcal{L}_{\mathrm{SbS}}(\mathcal{D}) = \sum_{i=1}^{N_{\mathcal{D}}} [\log p(y^{(i)} = w_{mn} | Q^{(i)}, \mathcal{T}^{(i)}) + \alpha \sum_{\ell=1}^{L} \log \tilde{w}(\mathbf{f}^{\star}_{k,\ell}, \cdot, \cdot)]$$



Experimental Setups

- Synthetic dataset containing 4 types of queries generated by templates

N.B. Natural Language with Templates <==> Formal language w/ or w/o ambiguity

Neural Enquirer is a kind of Pseudo Compiling

Our synthetic dataset consists of query-table-answer triples $\{(Q^{(i)}, \mathcal{T}^{(i)}, y^{(i)})\}$. To generate such a triple, we first randomly sample a table $\mathcal{T}^{(i)}$ of size 10×10 from a synthetic schema of Olympic Games, which has 10 fields, whose values are drawn from a vocabulary of size 240, with 120 country and city names, and 120 numbers. Figure $\boxed{5}$ gives an example table with one row. Next, we generate a query $Q^{(i)}$ using predefined templates associated with its gold-standard answer $y^{(i)}$ on $\mathcal{T}^{(i)}$.

Query Type	Example Queries with Annotated SQL-like Logical Forms
SELECT_WHERE	▷ Q_1: *How many people participated in the game in Beijing?* F_1: `select #_participants, where host_city = Beijing` ▷ Q_2: *In which country was the game hosted in 2012?* F_2: `select host_country, where year = 2012`
SUPERLATIVE	▷ Q_3: *When was the lastest game hosted?* F_3: `argmax(host_city, year)` ▷ Q_4: *How big is the country which hosted the shortest game?* F_4: `argmin(country_size, #_duration)`
WHERE_SUPERLATIVE	▷ Q_5: *How long is the game with the most medals that has fewer than 3,000 participants?* F_5: `where #_participants < 3,000, argmax(#_duration, #_medals)` ▷ Q_6: *How many medals are in the first game after 2008?* F_6: `where #_year > 2008, argmin(#_medals, #_year)`
NEST	▷ Q_7: *Which country hosted the longest game before the game in Athens?* F_7: `where year<(select year,where host_city=Athens),argmax(host_country,#_duration)` ▷ Q_8: *How many people watched the earliest game that lasts for more days than the game in 1956?* F_8: `where #_duration<(select #_duration,where year=1956),argmin(#_audience,#_year)`

Quantitative Results

	(Baseline)	Mixtured-25K			Mixtured-100K		
	Sempre	N2N	SbS	N2N - OOV	N2N	SbS	N2N - OOV
Select_Where	93.8%	96.2%	99.7%	90.3%	99.3%	100.0%	97.6%
Superlative	97.8%	98.9%	99.5%	98.2%	99.9%	100.0%	99.7%
Where_Superlative	34.8%	80.4%	94.3%	79.1%	98.5%	99.8%	98.0%
Nest	34.4%	60.5%	92.1%	57.7%	64.7%	99.7%	63.9%
Overall Acc.	65.2%	84.0%	96.4%	81.3%	90.6%	99.9%	89.8%

Qualitative Analysis

Q_5: *How long is the game with the most medals that has fewer than 3,000 participants?*

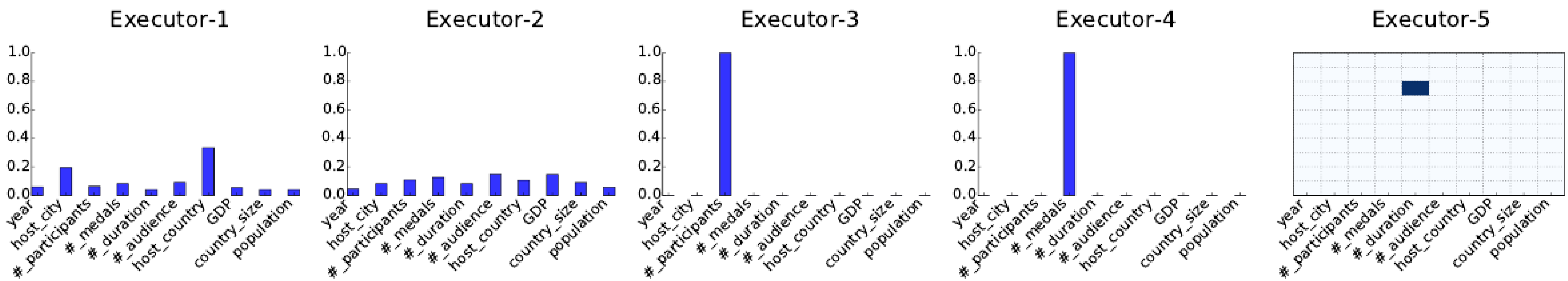


Figure 6: Weights visualization of query Q_5

- year
- host_city
- #_participants
- #_duration
- #_medals
- #_audience
- host_country
- GDP
- country_size
- population

Q_7: *Which country hosted the longest game before the game in Athens?*

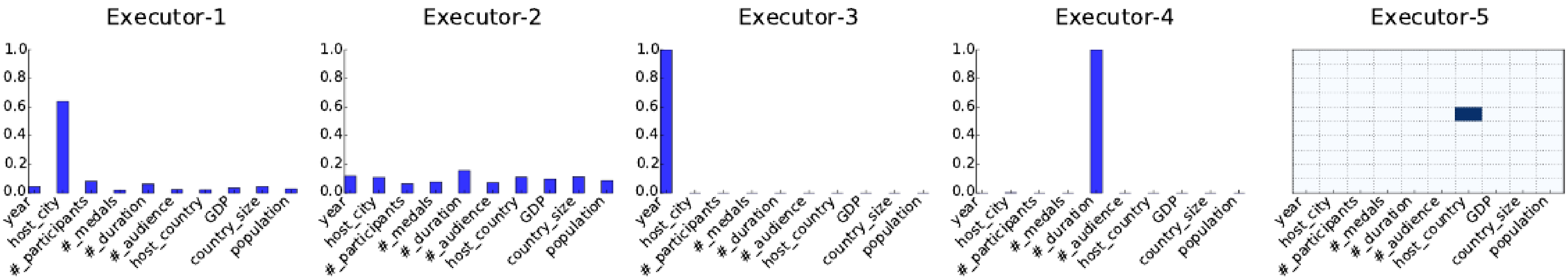


Figure 7: Weights visualization of query Q_7

- year
- host_city
- #_participants
- #_duration
- #_medals
- #_audience
- host_country
- GDP
- country_size
- population

Q_8: *How many people watched the earliest game that lasts for more days than the game in 1956?*

Figure 8: Weights visualization of query Q_8 (an incorrectly answered query)

- year
- host_city
- #_participants
- #_duration
- #_medals
- #_audience
- host_country
- GDP
- country_size
- population

Outline

- Neural Responding Machine
- Neural Question Answering
- Neural Enquirer: Learning to Query Tables with Natural Language
- Incorporating Copying Mechanism in Sequence-to-Sequence Learning

Incorporating Copying Mechanism in Sequence-to-Sequence Learning

Jiatao Gu[†] **Zhengdong Lu**[‡] **Hang Li**[‡] **Victor O.K. Li**[†]
[†]Department of Electrical and Electronic Engineering, The University of Hong Kong
{jiataogu, vli}@eee.hku.hk
[‡]Huawei Noah's Ark Lab, Hong Kong
{lu.zhengdong, hangli.hl}@huawei.com

arXiv 1603.06393

I:	Hello Jack, my name is Chandralekha.
R:	Nice to meet you, Chandralekha.
I:	This new guy doesn't perform exactly as we expected.
R:	What do you mean by "doesn't perform exactly as we expected"?

$$p(y_t|\mathbf{s}_t, y_{t-1}, \mathbf{c}_t, \mathbf{M}) = p(y_t, \mathsf{g}|\mathbf{s}_t, y_{t-1}, \mathbf{c}_t, \mathbf{M}) + p(y_t, \mathsf{c}|\mathbf{s}_t, y_{t-1}, \mathbf{c}_t, \mathbf{M}) \quad (4)$$

where g stands for the generate-mode, and c the copy mode. The probability of the two modes are given respectively by

$$p(y_t, \mathsf{g}|\cdot) = \begin{cases} \frac{1}{Z} e^{\psi_g(y_t)}, & y_t \in \mathcal{V} \\ 0, & y_t \in \mathcal{X} \cap \bar{V} \\ \frac{1}{Z} e^{\psi_g(\text{UNK})} & y_t \notin \mathcal{V} \cup \mathcal{X} \end{cases} \quad (5)$$

$$p(y_t, \mathsf{c}|\cdot) = \begin{cases} \frac{1}{Z} \sum_{j:x_j = y_t} e^{\psi_c(x_j)}, & y_t \in \mathcal{X} \\ 0 & \text{otherwise} \end{cases} \quad (6)$$

where $\psi_g(\cdot)$ and $\psi_c(\cdot)$ are score functions for generate-mode and copy-mode, respectively, and Z is the normalization term shared by the two modes, $Z = \sum_{v \in \mathcal{V} \cup \{\text{UNK}\}} e^{\psi_g(v)} + \sum_{x \in X} e^{\psi_c(x)}$.

s: state

M: {h1, .. ht}, i.e., source's states

c: input context

(w/ attent.)

Cf: conditional probability

Micro avg. vs. Macro avg.

Cf: softmax w/ multiple input

Mean field approximation

$$p = .5\left(\frac{\exp(z1)}{\sum \exp(z1')} + \frac{\exp(z2)}{\sum \exp(z2')}\right)$$

It's very hard to determine which one is better than another.

$$p(y_t|\mathbf{s}_t, y_{t-1}, \mathbf{c}_t, \mathbf{M}) = p(y_t, \mathsf{g}|\mathbf{s}_t, y_{t-1}, \mathbf{c}_t, \mathbf{M}) + p(y_t, \mathsf{c}|\mathbf{s}_t, y_{t-1}, \mathbf{c}_t, \mathbf{M}) \quad (4)$$

where g stands for the generat
copy mode. The probability of
given respectively by

$$\boxed{\psi_g(y_t = v_i) = \mathbf{v}_i^\top \mathbf{W}_o \mathbf{s}_t, \quad v_i \in \mathcal{V} \cup \text{UNK} \quad (7)}$$

where $\mathbf{W}_o \in \mathbb{R}^{(N+1)\times d_s}$ and $\mathbf{v}_i$ is the one-hot indicator vector for v_i.

$$p(y_t, \mathsf{g}|\cdot) = \begin{cases} \frac{1}{Z} e^{\psi_g(y_t)}, & y_t \in \mathcal{V} \\ 0, & y_t \in \mathcal{X} \cap \bar{V} \\ \frac{1}{Z} e^{\psi_g(\text{UNK})} & y_t \notin \mathcal{V} \cup \mathcal{X} \end{cases} \quad (5)$$

$$p(y_t, \mathsf{c}|\cdot) = \begin{cases} \frac{1}{Z} \sum_{j:x_j = y_t} e^{\psi_c(x_j)}, & y_t \in \mathcal{X} \\ 0 & \text{otherwise} \end{cases} \quad (6)$$

where $\psi_g(\cdot)$ and $\psi_c(\cdot)$ are score functions for generate-mode and copy-mode, respectively, and Z is the normalization term shared by the two modes, $Z = \sum_{v \in \mathcal{V} \cup \{\text{UNK}\}} e^{\psi_g(v)} + \sum_{x \in X} e^{\psi_c(x)}$.

vW: weight matrix indexed by word i
s: hidden state
→ Nothing but linear transformation

$$p(y_t|\mathbf{s}_t, y_{t-1}, \mathbf{c}_t, \mathbf{M}) = p(y_t, \mathsf{g}$$
$$+ p(y_t, \mathsf{c}|\mathbf{s}_t,$$

where g stands for the generate
copy mode. The probability of t
given respectively by

$$p(y_t, \mathsf{g}|\cdot)=\begin{cases} \frac{1}{Z} e^{\psi_g(y_t)}, \\ 0, \\ \frac{1}{Z} e^{\psi_g(\text{UNK})} \end{cases}$$

$$p(y_t, \mathsf{c}|\cdot)=\begin{cases} \frac{1}{Z} \sum_{j:x_j=y_t} e^{\psi_c(x_j)}, & y_t \in \mathcal{X} \\ 0 & \text{otherwise} \end{cases} \quad (6)$$

where $\psi_g(\cdot)$ and $\psi_c(\cdot)$ are score functions for generate-mode and copy-mode, respectively, and Z is the normalization term shared by the two modes, $Z = \sum_{v \in \mathcal{V} \cup \{\text{UNK}\}} e^{\psi_g(v)} + \sum_{x \in X} e^{\psi_c(x)}$.

Copy-Mode: The score for "copying" the word x_j is calculated as

$$\psi_c(y_t = x_j) = \sigma\left(\mathbf{h}_j^\top \mathbf{W}_c\right) \mathbf{s}_t, \quad x_j \in \mathcal{X} \quad (8)$$

where $\mathbf{W}_c \in \mathbb{R}^{d_h \times d_s}$, and σ is an activation that is either an identity or a non-linear function such as $\tanh$. When calculating the copy-mode score, we use the hidden states $\{\mathbf{h}_1, ..., \mathbf{h}_{T_S}\}$ to "represent" each of the word in the source sequence $\{x_1, ..., x_{T_S}\}$ since the bi-directional RNN encodes not only the content, but also the location information into the hidden states in $\mathbf{M}$.

State Update (Input)

- $[\mathbf{e}(y_{t-1}); \zeta(y_{t-1})]^\top$
- **e**(): embedding of a word
- zeta(): $\zeta(y_{t-1}) = \sum_{\tau=1}^{T_S} \rho_{t\tau} \mathbf{h}_\tau$

If the last word is copied from x_t

$$\rho_{t\tau} = \begin{cases} \frac{1}{K} p(x_\tau, \mathsf{c}|\mathbf{s}_{t-1}, \mathbf{M}), & x_\tau = y_{t-1} \\ 0 & \text{otherwise} \end{cases}$$

$$K = \sum_{\tau' : x_{\tau'} = y_{t-1}} p(x_{\tau'}, c|\mathbf{s}_{t-1}, \mathbf{M})$$

I don't see formal definition of *p* , but it shall be similar to attent.

$$\mathbf{c}_t = \sum_{\tau=1}^{T_S} \alpha_{t\tau} \mathbf{h}_\tau; \quad \alpha_{t\tau} = \frac{e^{\eta(\mathbf{s}_{t-1}, \mathbf{h}_\tau)}}{\sum_{\tau'} e^{\eta(s_{t-1}, \mathbf{h}_{\tau'})}}$$

State Update (Input)

- $\left[\mathbf{e}(y_{t-1}); \zeta(y_{t-1})\right]^\top$
- **e**(): embedding of a word
- zeta(): $\zeta(y_{t-1}) = \sum_{\tau=1}^{T_S} \rho_{t\tau}\mathbf{h}_\tau$

Local-based Addressing (good for OOV)

$\zeta(y_{t-1}) \xrightarrow{\text{update}} \mathbf{s}_t \xrightarrow{\text{predict}} y_t \xrightarrow{\text{sel. read}} \zeta(y_t)$

If the last word is copied from x_t

$$\rho_{t\tau} = \begin{cases} \frac{1}{K} p(x_\tau, \mathsf{c}|\mathbf{s}_{t-1}, \mathbf{M}), & x_\tau = y_{t-1} \\ 0 & \text{otherwise} \end{cases}$$

$$K = \sum_{\tau' : x_{\tau'} = y_{t-1}} p(x_{\tau'}, c|\mathbf{s}_{t-1}, \mathbf{M})$$

Learning

- End-to-end fashion

$$\mathcal{L} = -\frac{1}{N}\sum_{k=1}^{N}\sum_{t=1}^{T}\log\left[p(y_t^{(k)}|y_{<t}^{(k)}, X^{(k)})\right]$$

Discussion

- Designing highly (more and more) complicated neural networks to mimic human behaviors: modeling a sentence, querying a table/KB, selecting a field/column, selecting a row, copying something, etc.)
- The network has been somewhat over-complicated; it is very hard to judge which part actually contributes to the performance.
- Evaluation is oftentimes weak: synthetic data, subjective evaluation, or criterion not clear (e.g., genQA), etc.
- Nevertheless, an important school of DL4NLP.

A Wider Scope

- Learning to Execute
- Neural Programmer
- Neural Program Interpreter
- Latent Predictor Network for Code Generation

Challenge of end-to-end learning:

- Information processing

	avg	sum	max	attention	argmax
Differentiability	😋	😋	😋	😋	😿
Supervision	😐	😐	😐	😐	😿
Scalability	😿	😿	😿	😿	😋

Intuition

- Using external information to guide an NN instead of designing end-to-end machines
  - Better performance in short term
  - May or may not conform to the goal of AI, depending on how strict the external information is

	Hard mechanism
Differentiability	😋
Supervision	😋
Scalability	😋

(e.g., if-statement)