Neural Symbolic Machines

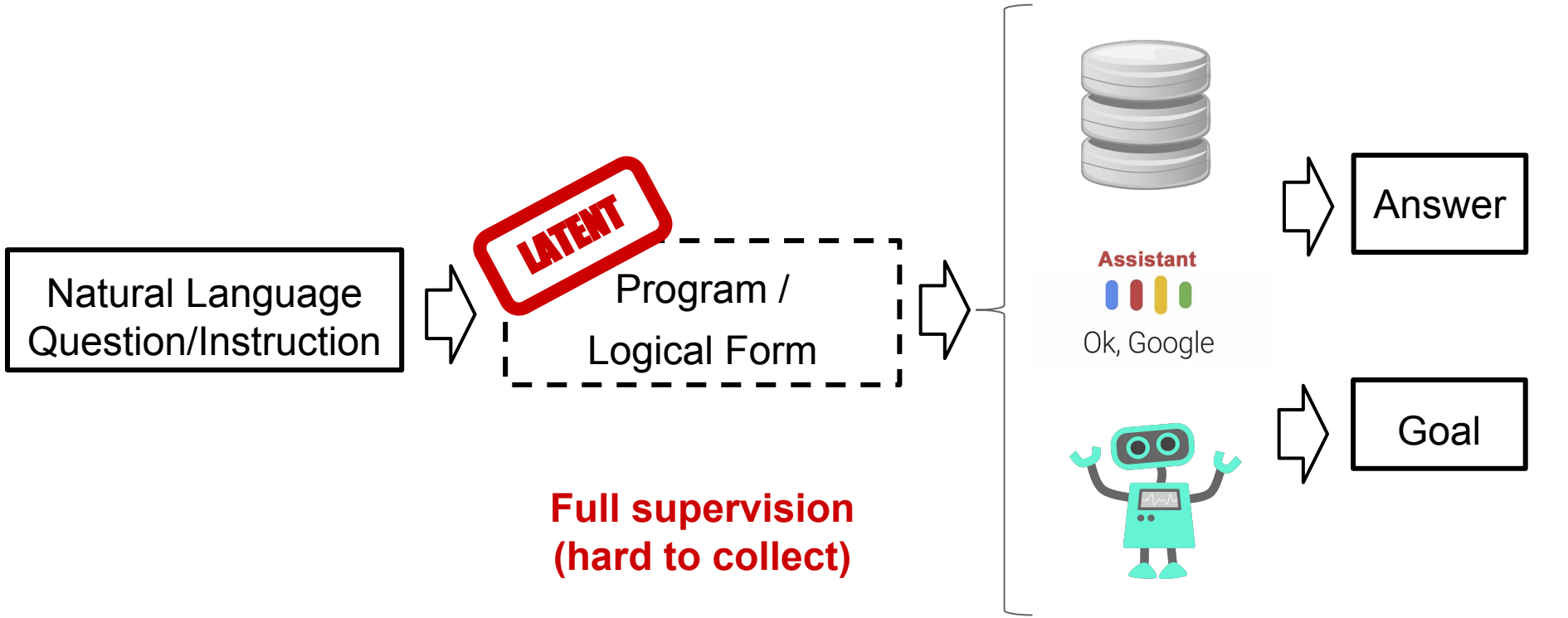
Semantic Parsing on Freebase with Weak Supervision

Chen Liang, Jonathan Berant, Quoc Le, Kenneth Forbus, Ni Lao

Overview

- Motivation: Semantic Parsing and Program Induction
- Neural Symbolic Machines
  - Key-Variable Memory
  - Code Assistance
  - Augmented REINFORCE
- Experiments and analysis

Semantic Parsing: Language to Programs



Full supervision (hard to collect)

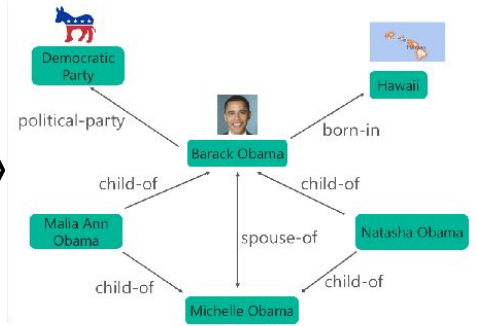
Weak supervision (easy to collect)

[Berant, et al 2013; Liang 2013]

Question Answering with Knowledge Base

Largest city in US?

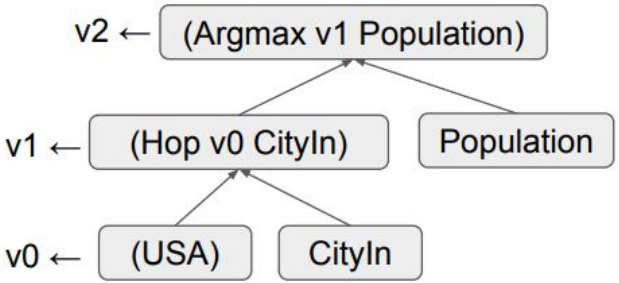
```
GO
(Hop V1 CityIn)
(Argmax V2 Population)
RETURN
```



NYC

Freebase, DBpedia, YAGO, NELL

1. Compositionality



2. Large Search Space

Freebase:
23K predicates, 82M entities, 417M triplets

WebQuestionsSP Dataset

- 5,810 questions Google Suggest API & Amazon MTurk[1]
- Remove invalid QA pairs[2]
- 3,098 training examples, 1,639 testing examples remaining
- Open-domain, and contains grammatical error
- Multiple entities as answer => macro-averaged F1

Grammatical error

- What do Michelle Obama do for a living?
- What character did Natalie Portman play in Star Wars?
- What currency do you use in Costa Rica?
- What did Obama study in school?
- What killed Sammy Davis Jr?

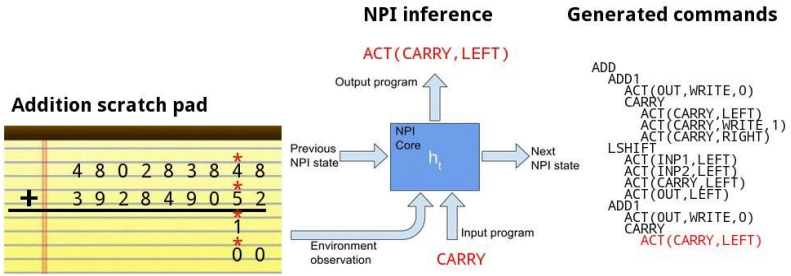
Multiple entities

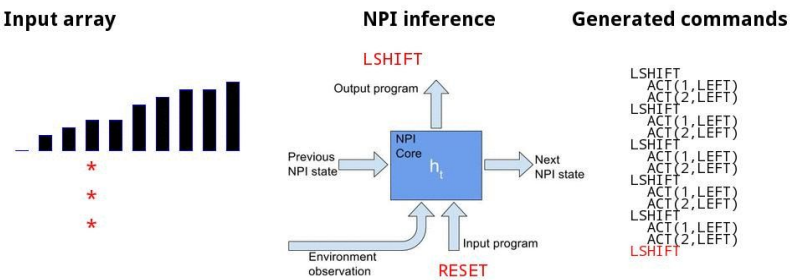
writer, lawyer
Padme Amidala
Costa Rican colon
political science
throat cancer

[Berant et al, 2013; Yih et al, 2016]

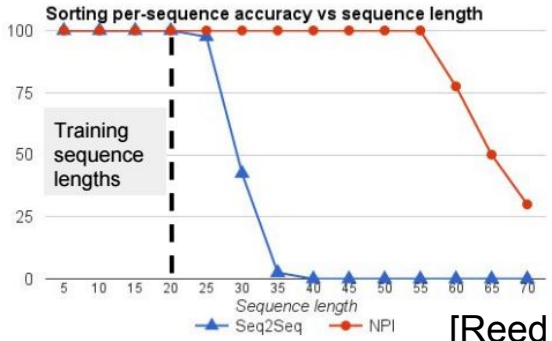
(Scalable) Neural Program Induction

- Impressive works to show NN can learn addition and sorting, but...



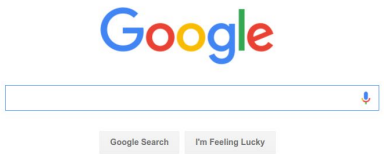


- The learned operations are not as scalable and precise.



[Reed & Freitas 2015]

- Why not use existing modules that are scalable, precise and interpretable?

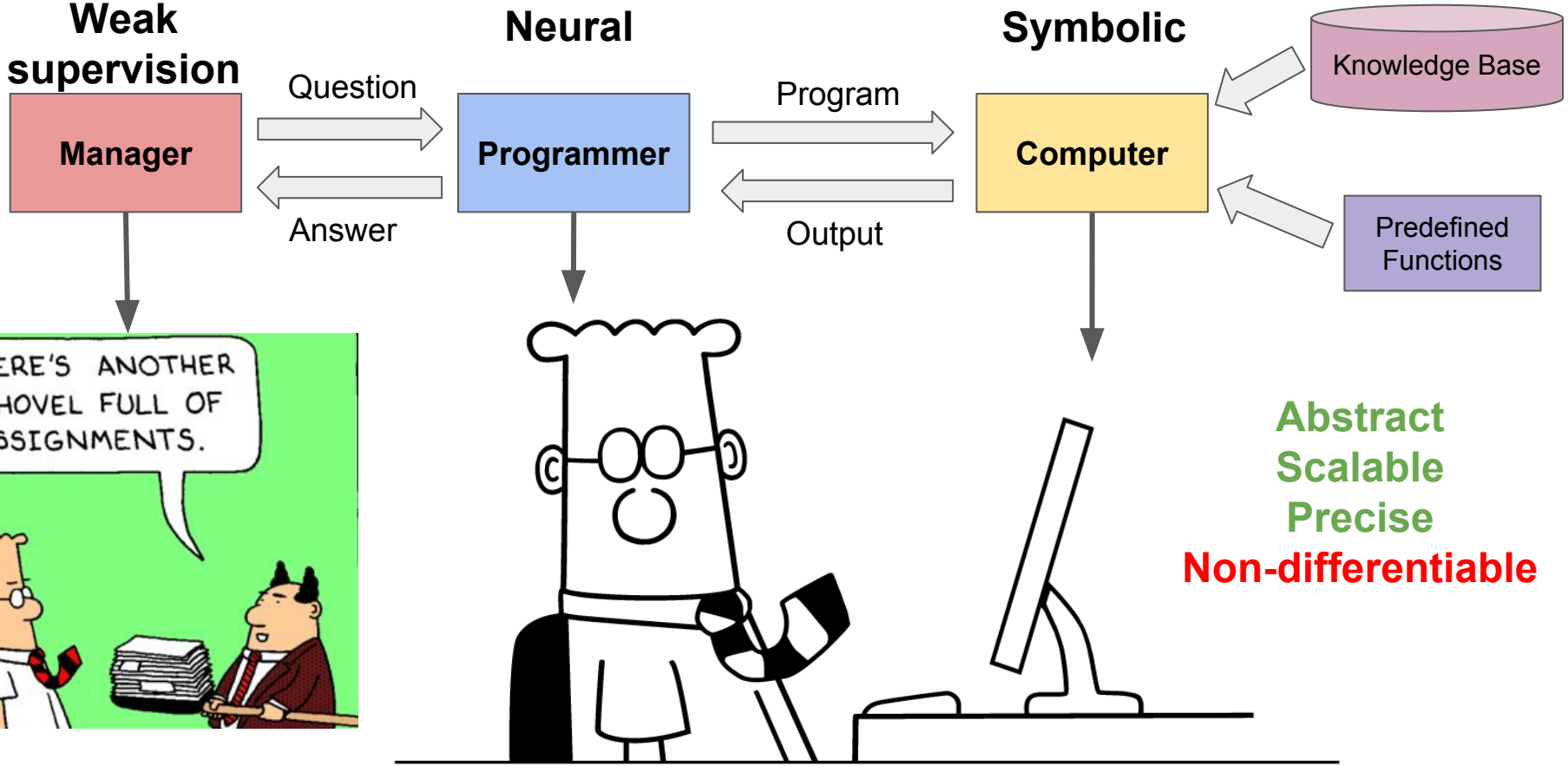


[Zaremba & Sutskever 2016]

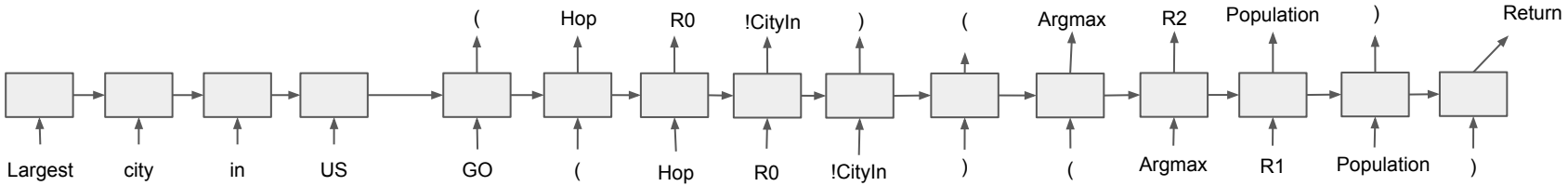
Overview

- Motivation: Semantic Parsing and Program Induction
- Neural Symbolic Machines
  - Key-Variable Memory
  - Code Assistance
  - Augmented REINFORCE
- Experiments and analysis

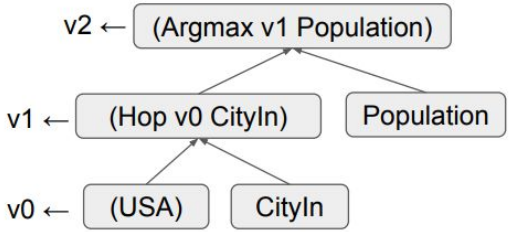
Neural Symbolic Machines



Simple Seq2Seq model is not enough



1. Compositionality



2. Large Search Space

23K predicates,
82M entities,
417M triplets

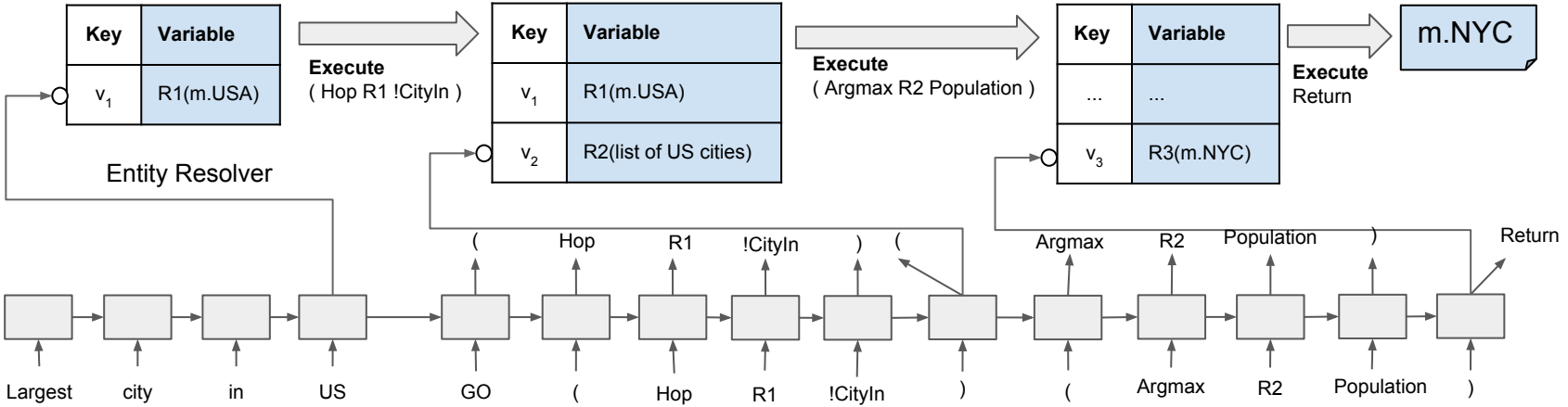
1.Key-Variable Memory

2.Code Assistance
3.Augmented REINFORCE

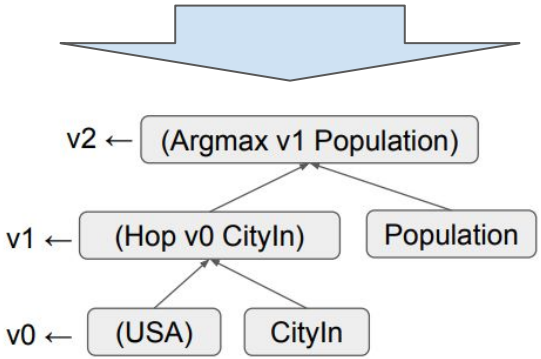
Overview

- Motivation: Semantic Parsing and Program Induction
- Neural Symbolic Machines
  - Key-Variable Memory
  - Code Assistance
  - Augmented REINFORCE
- Experiments and analysis

Key-Variable Memory for Compositionality



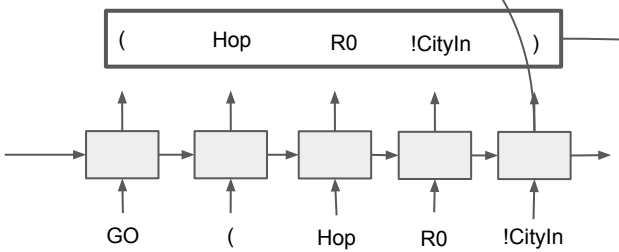
- A linearised bottom-up derivation of the recursive program.



Key-Variable Memory: Save Intermediate Value

Key (Embedding)	**Variable** (Symbol)	**Value** (Data in Computer)
V_0	`R0`	`m.USA`
V_1	`R1`	`[m.SF, m.NYC, ...]`

Expression is finished.



Execution

Computer

Result

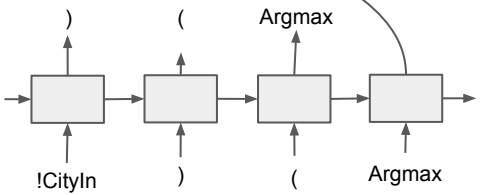
Key-Variable Memory: Reuse Intermediate Value

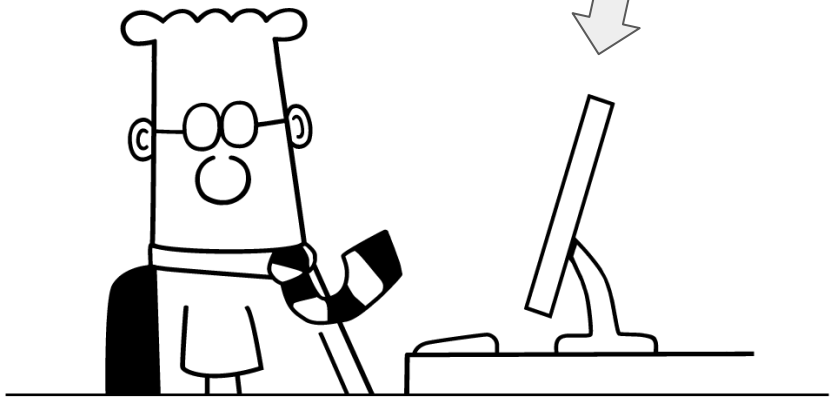
Key (Embedding)	**Variable** (Symbol)	**Value** (Data in Computer)
V_0	`R0`	`m.USA`
V_1	`R1`	`[m.SF, m.NYC, ...]`

Softmax

Neural

Symbolic



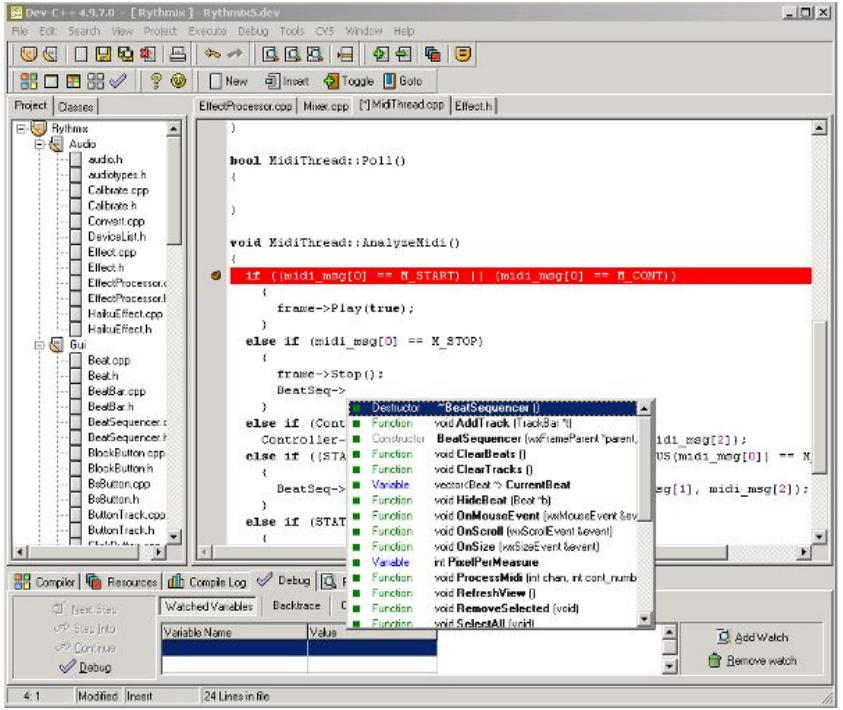


Overview

- Motivation: Semantic Parsing and Program Induction
- Neural Symbolic Machines
  - Key-Variable Memory
  - Code Assistance
  - Augmented REINFORCE
- Experiments and analysis

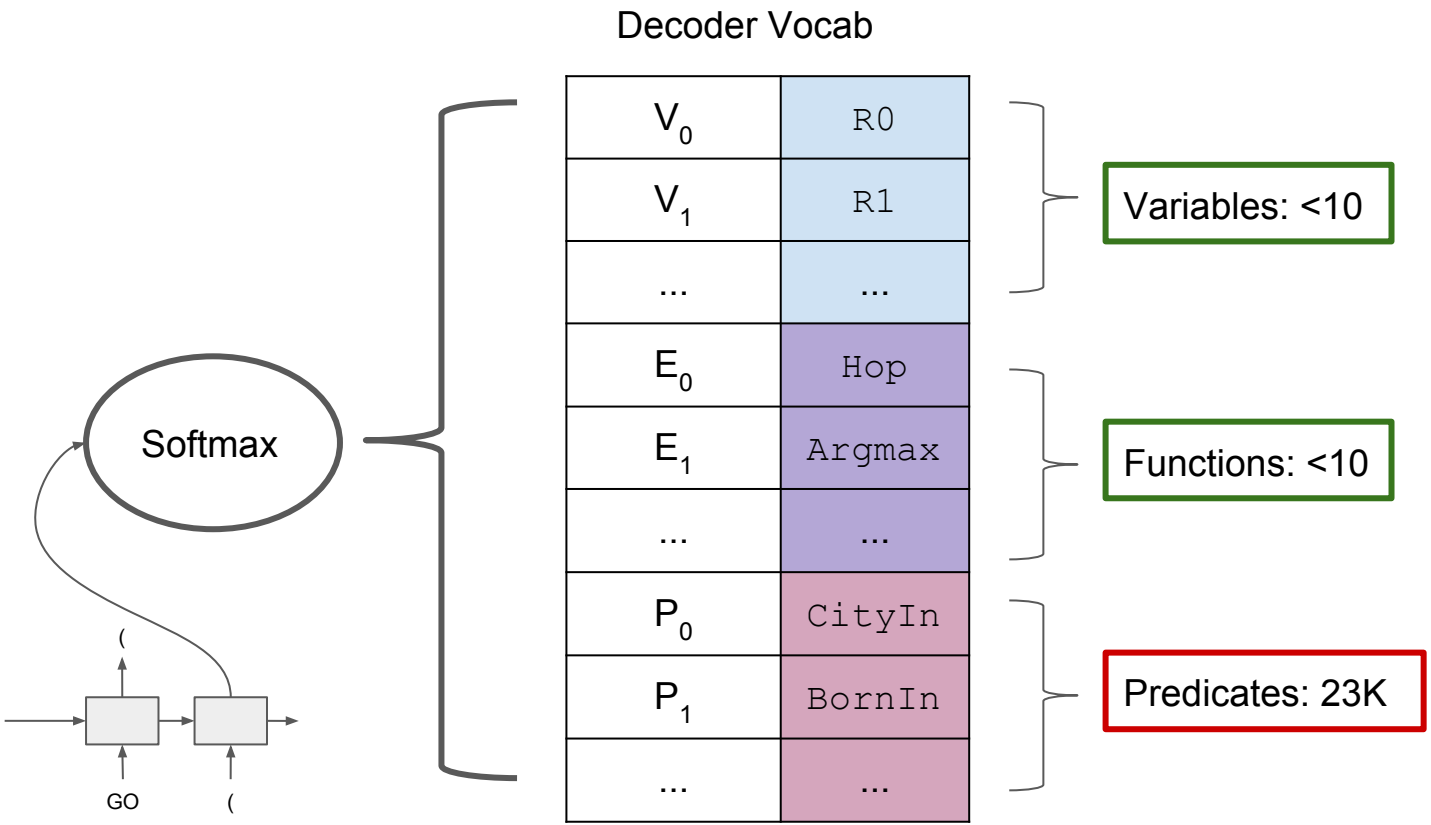
Code Assistance: Prune Search Space

Pen and paper



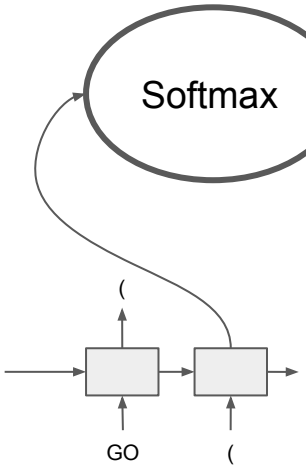
IDE

Code Assistance: Syntactic Constraint



Code Assistance: Syntactic Constraint

Last token is ‘(’, so has to output a function name next.



Decoder Vocab

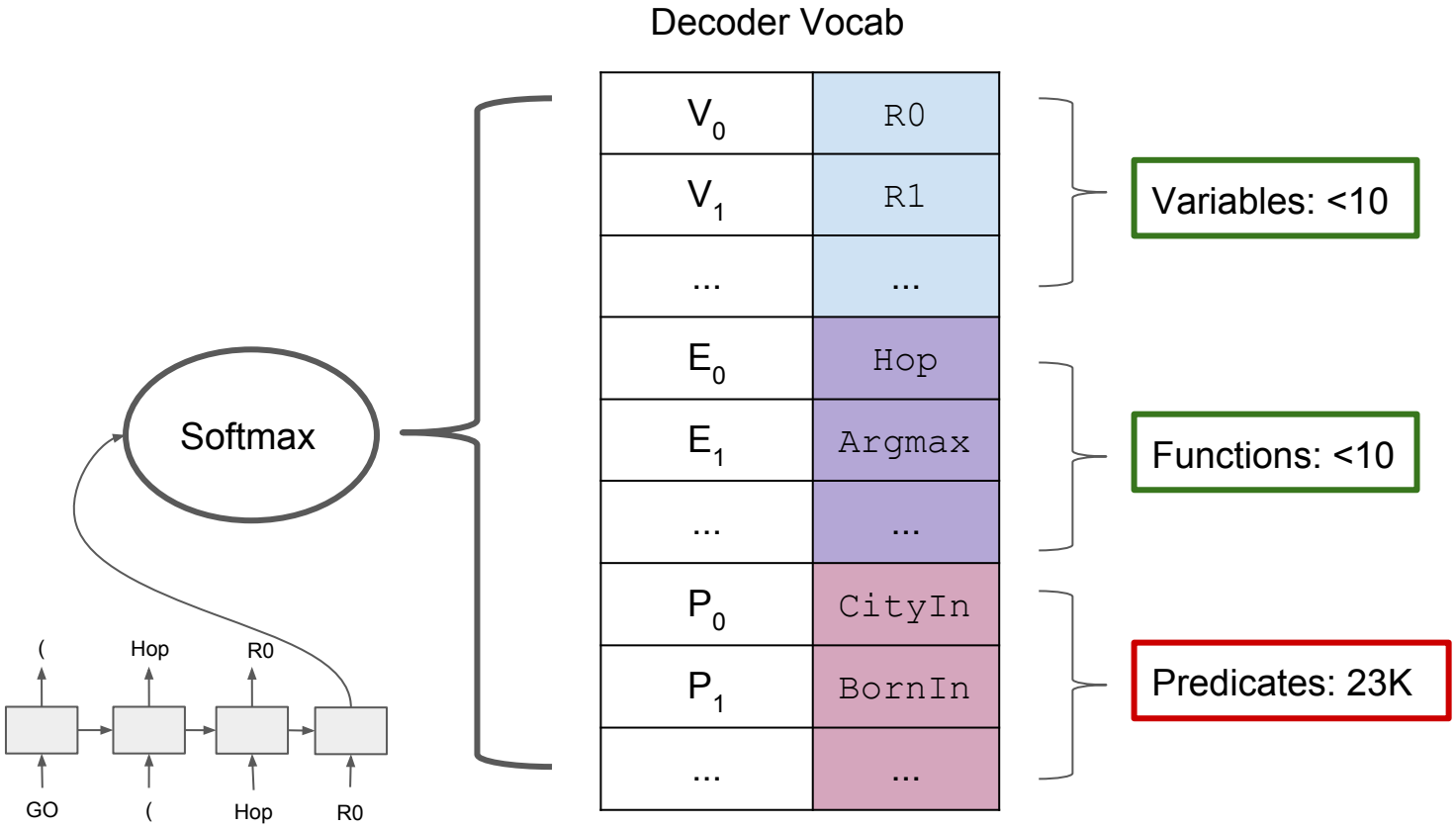
E_0	Hop
E_1	Argmax
...	...

Variables: <10

Functions: <10

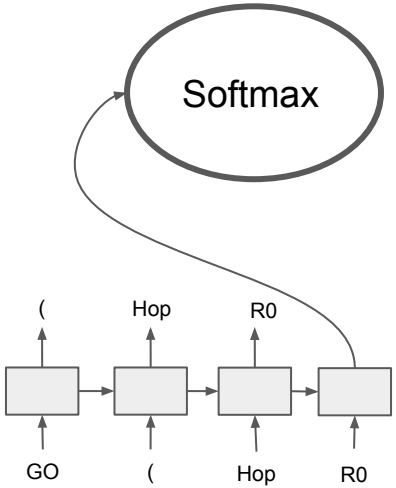
Predicates: 23K

Code Assistance: Semantic Constraint

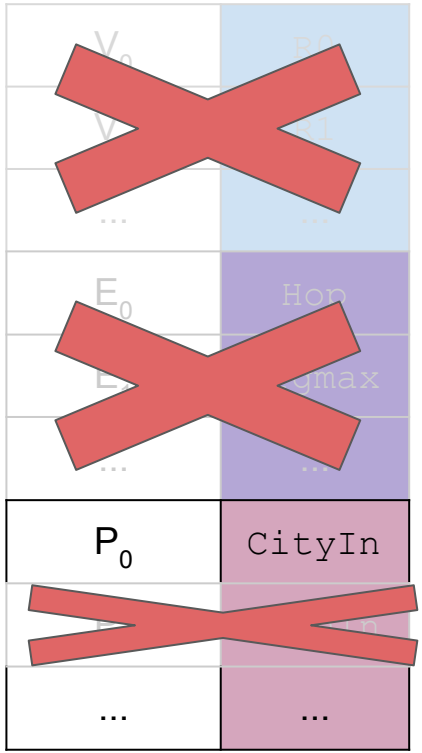


Code Assistance: Semantic Constraint

Given definition of `Hop`, need to output a predicate that is connected to `R2` (`m.USA`).



Decoder Vocab



Variables: <10

Functions: <10

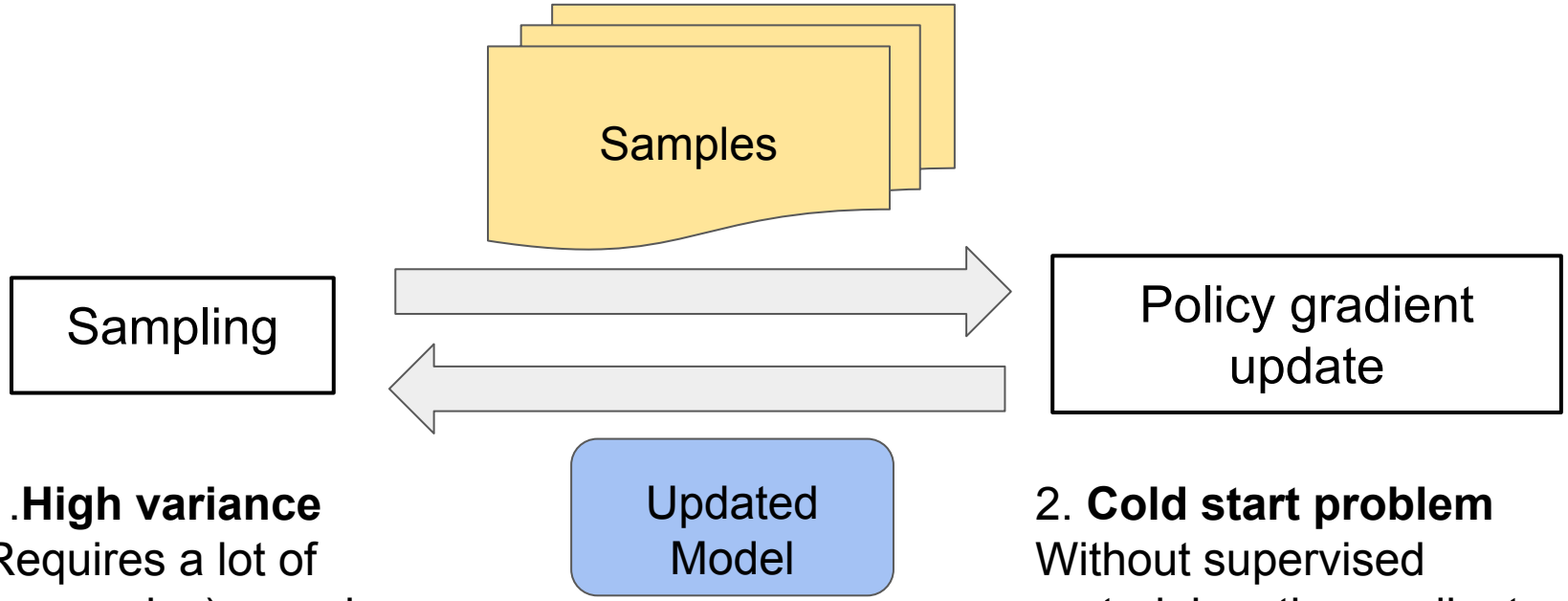
Predicates: 23K

Valid Predicates: <100

Overview

- Motivation: Semantic Parsing and Program Induction
- Neural Symbolic Machines
  - Key-Variable Memory
  - Code Assistance
  - Augmented REINFORCE
- Experiments and analysis

REINFORCE Training

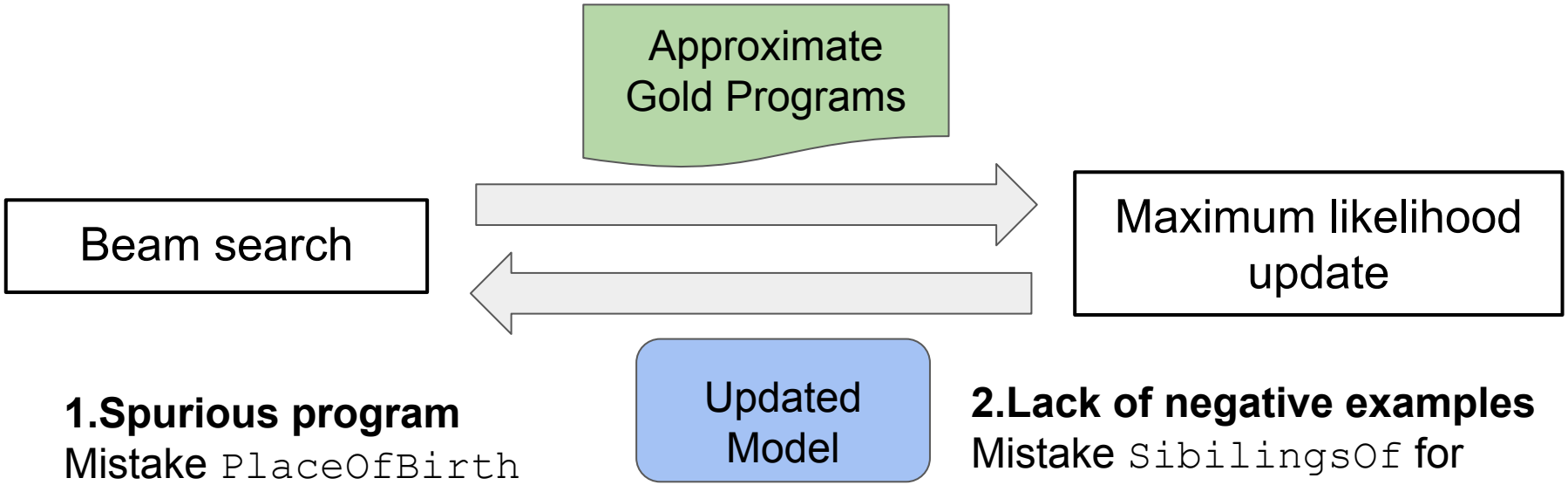


1.**High variance**
Requires a lot of (expensive) samples

2. **Cold start problem**
Without supervised pretraining, the gradients at the beginning

$$\nabla_\theta J^{RL}(\theta) = \sum_q \sum_{a_{0:T}} P(a_{0:T}|q,\theta)[R(q,a_{0:T}) - B(q)]\nabla_\theta \log P(a_{0:T}|q,\theta)$$

Iterative Maximum Likelihood Training (Hard EM)

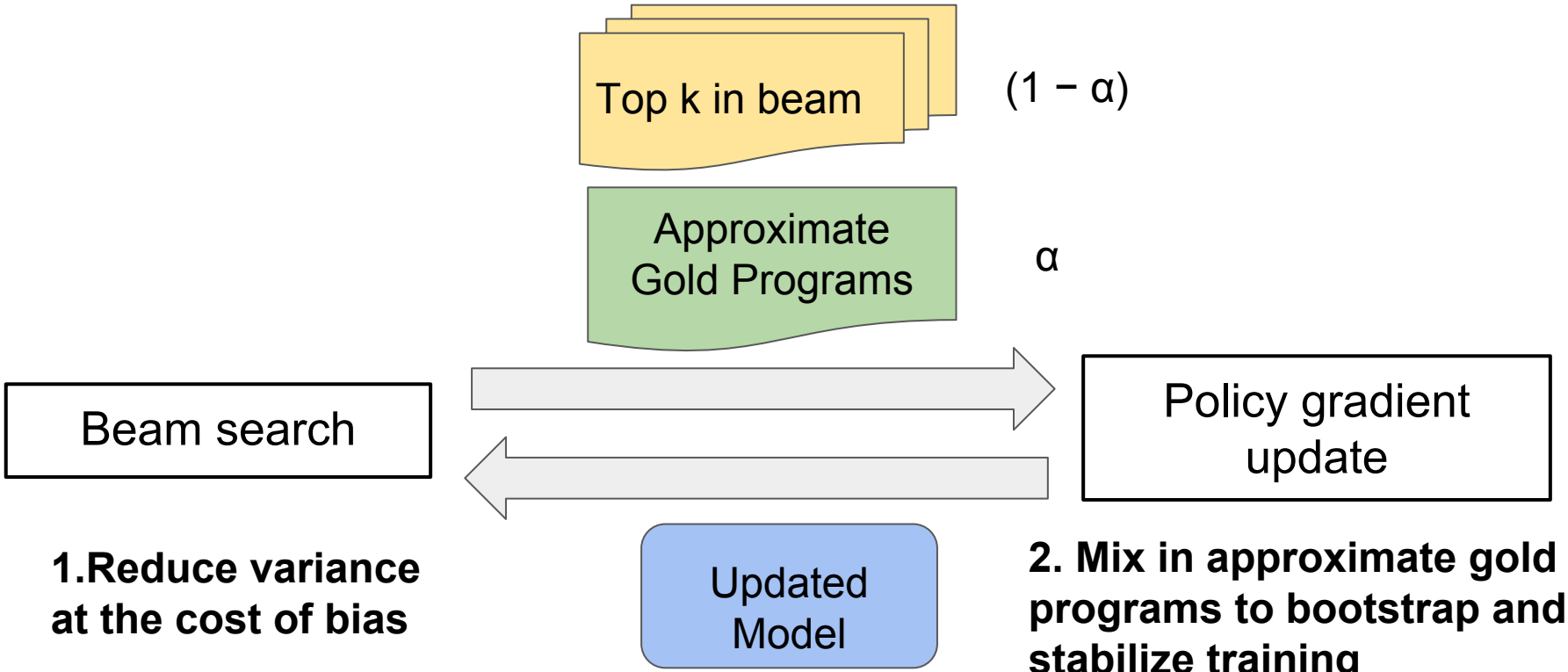


1.Spurious program
Mistake `PlaceOfBirth` for `PlaceOfDeath`.

2.Lack of negative examples
Mistake `SibilingsOf` for `ParentsOf`.

$$J^{ML}(\theta) = \sum_{q} \log P(a^{best}_{0:T}(q)|q,\theta)$$

Augmented REINFORCE



1.Reduce variance at the cost of bias

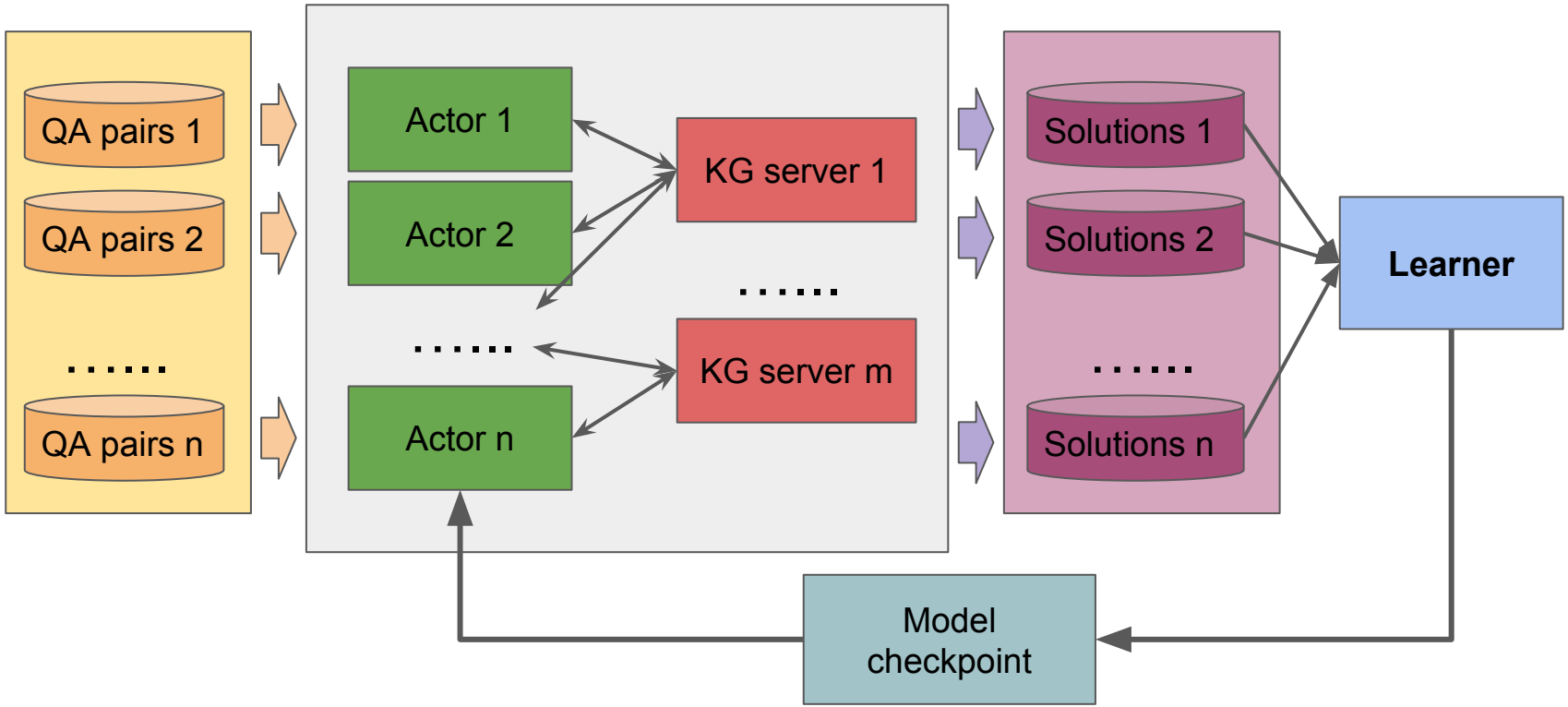
2. Mix in approximate gold programs to bootstrap and stabilize training

Overview

- Motivation: Semantic Parsing and Program Induction
- Neural Symbolic Machines
  - Key-Variable Memory
  - Code Assistance
  - Augmented REINFORCE
- Experiments and analysis

Distributed Architecture

- 200 actors, 1 learner, 50 Knowledge Graph servers



Generated Programs

- **Question**: “what college did russell wilson go to?”
- **Generated program**:

```
(hop v1 /people/person/education)
(hop v2 /education/education/institution)
(filter v3 v0 /common/topic/notable_types )
<EOP>
```

In which

```
v0 = “College/University” (m.01y2hnl)
v1 = “Russell Wilson” (m.05c10yf)
```

- **Distribution of the length of generated programs**

#Expressions	0	1	2	3
Percentage	0.4%	62.9%	29.8%	6.9%
F1	0.0	73.5	59.9	70.3

New State-of-the-Art on ***WebQuestionsSP***

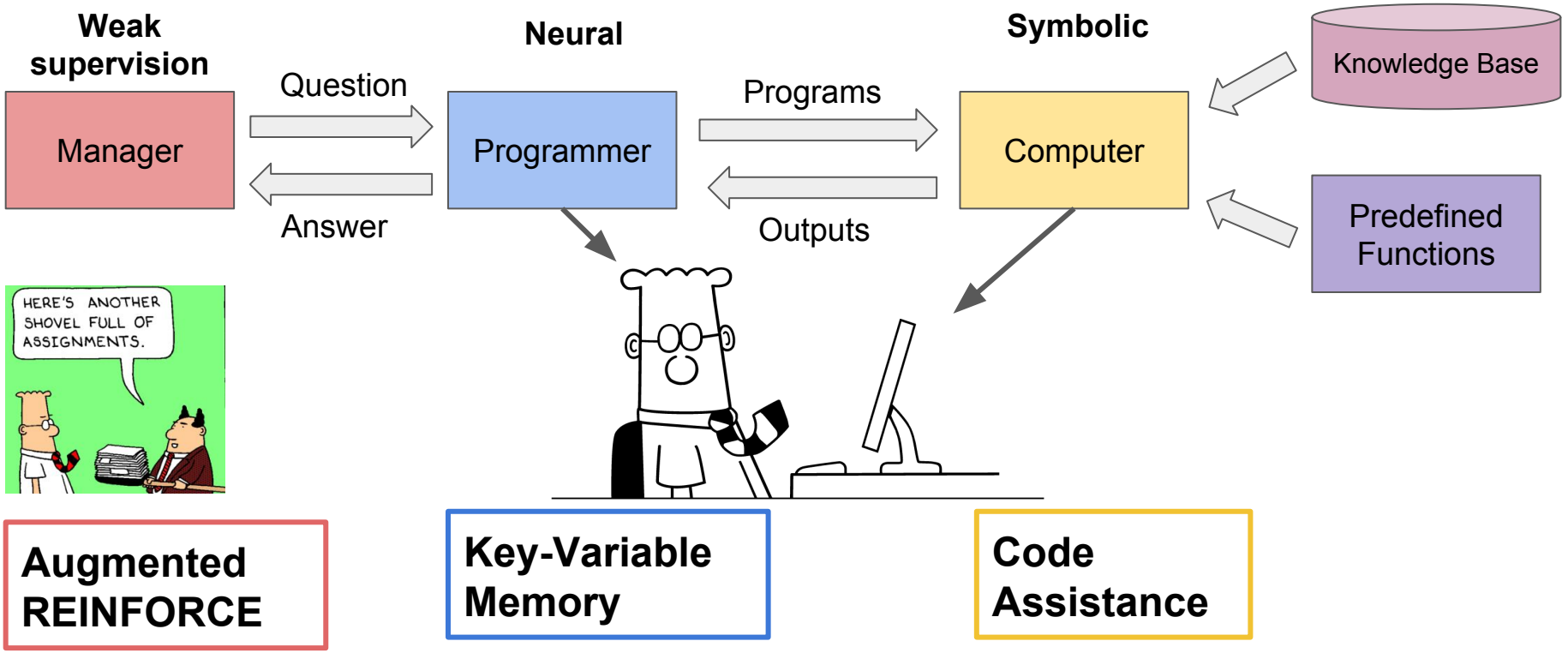
- First end-to-end neural network to achieve SOTA on semantic parsing with weak supervision over large knowledge base
- The performance is approaching SOTA with full supervision

Model	Avg. Prec.@1	Avg. Rec.@1	Avg. F1@1	Acc.@1
STAGG	67.3	73.1	66.8	58.8
NSM – our model	70.8	76.0	**69.0**	59.5
STAGG (full supervision)	70.9	80.3	71.7	63.9

Augmented REINFORCE

- REINFORCE get stuck at local maxima
- Iterative ML training is not directly optimizing the F1 score
- Augmented REINFORCE obtains the best performances

Settings	Train Avg. F1@1	Valid Avg. F1@1
iterative ML only	68.6	60.1
REINFORCE only	55.1	47.8
Augmented REINFORCE	83.0	**67.2**

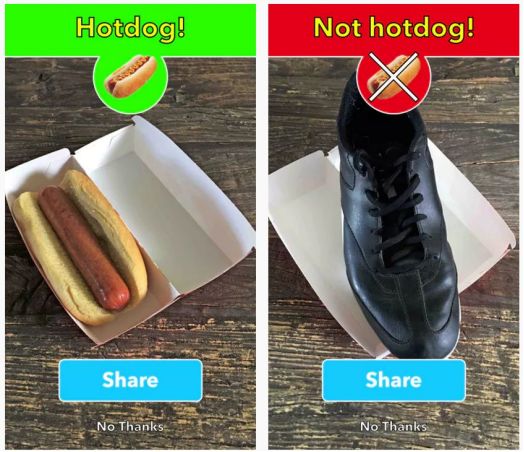

Weak supervision
Neural
Symbolic
Manager
Question
Answer
Programmer
Programs
Outputs
Computer
Knowledge Base
Predefined Functions
HERE'S ANOTHER SHOVEL FULL OF ASSIGNMENTS.
Augmented REINFORCE
Key-Variable Memory
Code Assistance

Thanks!

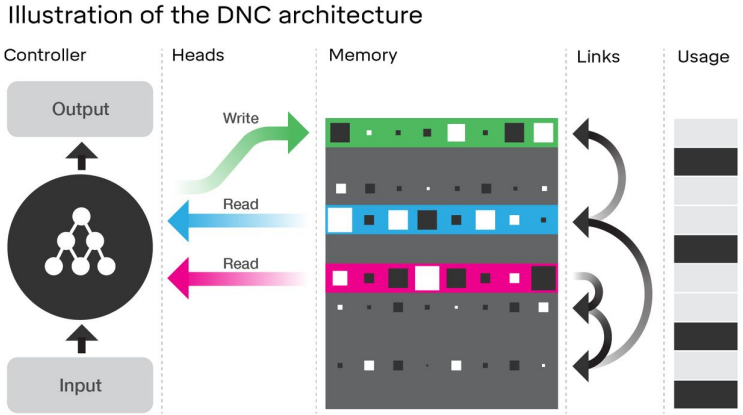
Backup Slides

Semantic Parsing as Program Induction

Learning classifiers



Learning programs

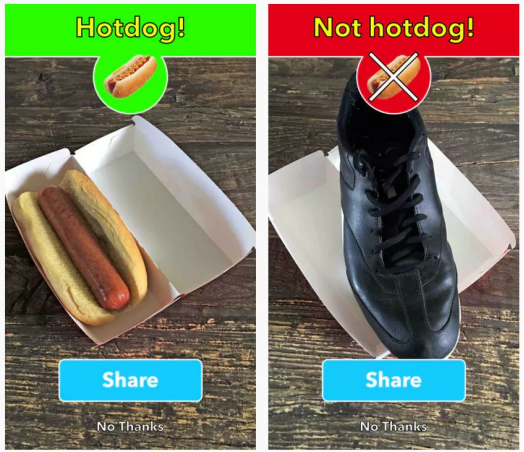


Semantic parsing: learning to write programs (given natural language instructions/questions)

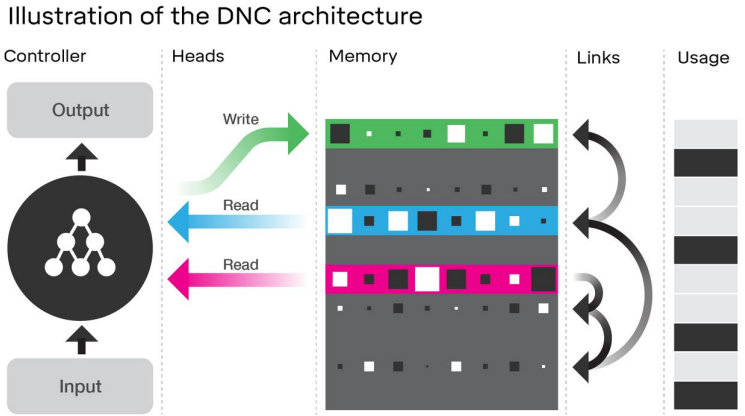
[Graves et al, 2016; Silicon Valley, Season 4]

Related Topic: Neural Program Induction

Learning classifiers



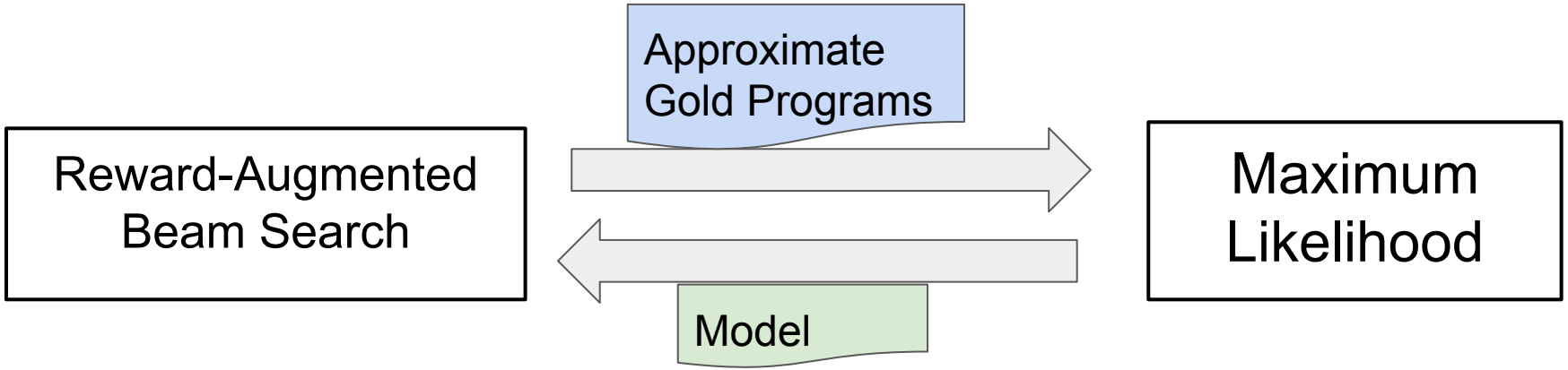
Learning programs



Semantic parsing: learning to write programs (given natural language instructions/questions)

[Graves et al, 2016; Silicon Valley, Season 4]

Iterative Maximum Likelihood Training

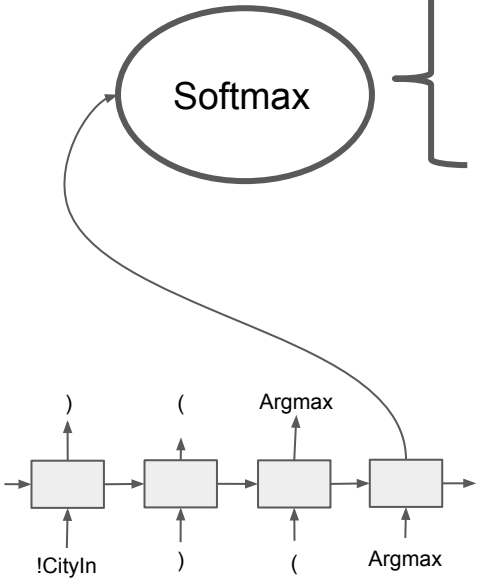


1.Spurious program
Mistake `PlaceOfBirth` for `PlaceOfDeath`.

2.Lack of negative examples
Mistake `SibilingsOf` for `ParentsOf`.

$$J^{ML}(\theta) = \sum_{q} \log P(a_{0:T}^{best}(q)|q,\theta)$$

Key-Variable Memory: Reuse Intermediate Value



Key (Embedding)	**Variable** (Symbol)	**Value** (Data in Computer)
V_0	`R0`	`m.USA`
V_1	`R1`	`[m.SF, m.NYC, ...]`

Generated Programs

- **Question**: “what college did russell wilson go to?”
- **Generated program**:

```
(hop v1 /people/person/education)
(hop v2 /education/education/institution)
(filter v3 v0 /common/topic/notable_types )
<EOP>
```

In which

```
v0 = “College/University” (m.01y2hnl)
v1 = “Russell Wilson” (m.05c10yf)
```

- **Distribution of the length of generated programs**

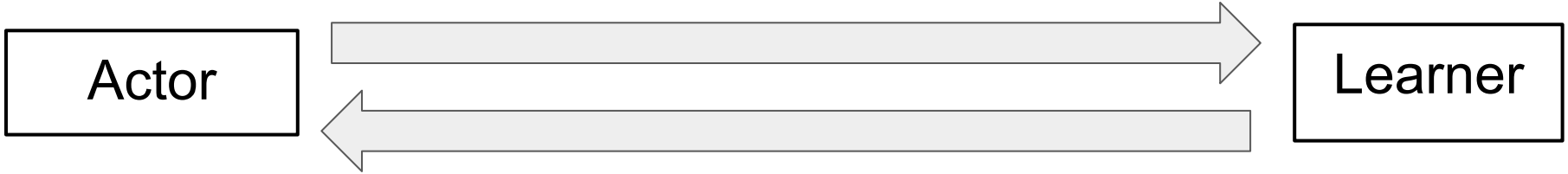
#Expressions	0	1	2	3
Percentage	0.4%	62.9%	29.8%	6.9%
F1	0.0	73.5	59.9	70.3

REINFORCE

Repeat

1.**High variance**
Requires a lot of (expensive) samples

Sampling



Policy gradient

$$\nabla_\theta J^{RL}(\theta) = \sum_q \sum_{a_{0:T}} P(a_{0:T}|q,\theta)[R(q,a_{0:T}) - B(q)]\nabla_\theta \log P(a_{0:T}|q,\theta)$$

2. **Bootstrap problem**
Small gradients at the beginning

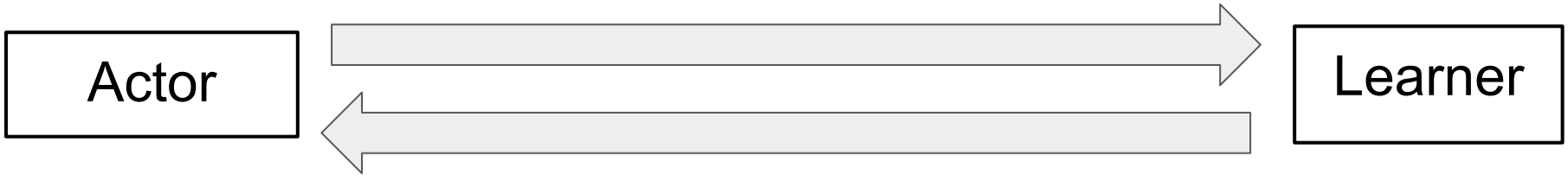
Samples

Iterative Maximum Likelihood Training

Repeat

1.Spurious program
Mistake `PlaceOfBirth` for `PlaceOfDeath`.

Reward-Augmented Beam Search



Maximum Likelihood

$$J^{ML}(\theta) = \sum_q \log P(a_{0:T}^{best}(q)|q,\theta)$$

Approximate Gold Programs

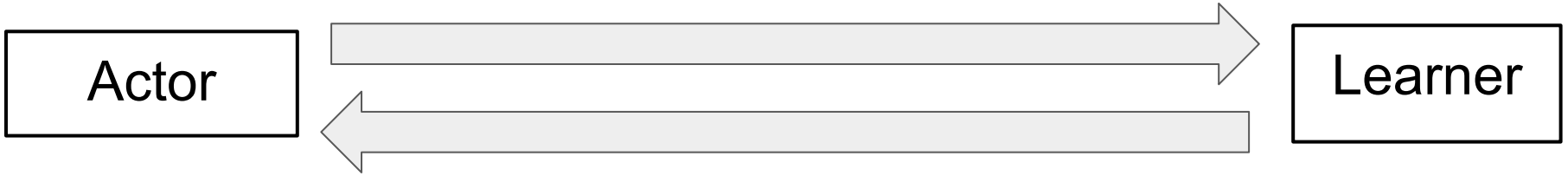
2.Lack of negative examples
Mistake `SibilingsOf` for `ParentsOf`.

Augmented REINFORCE

Repeat

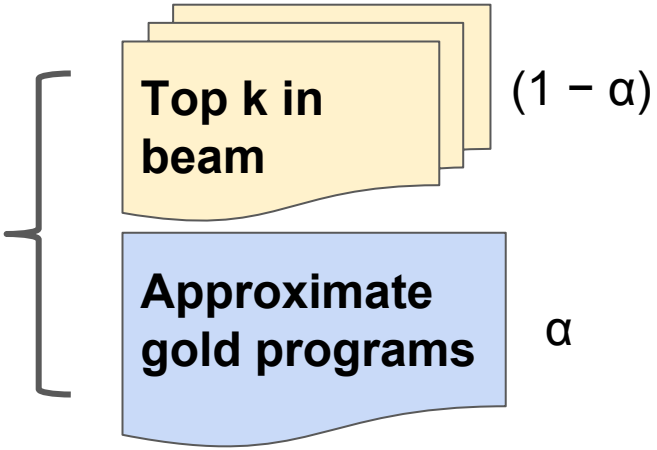
Beam Search

Reduce variance at the cost of bias

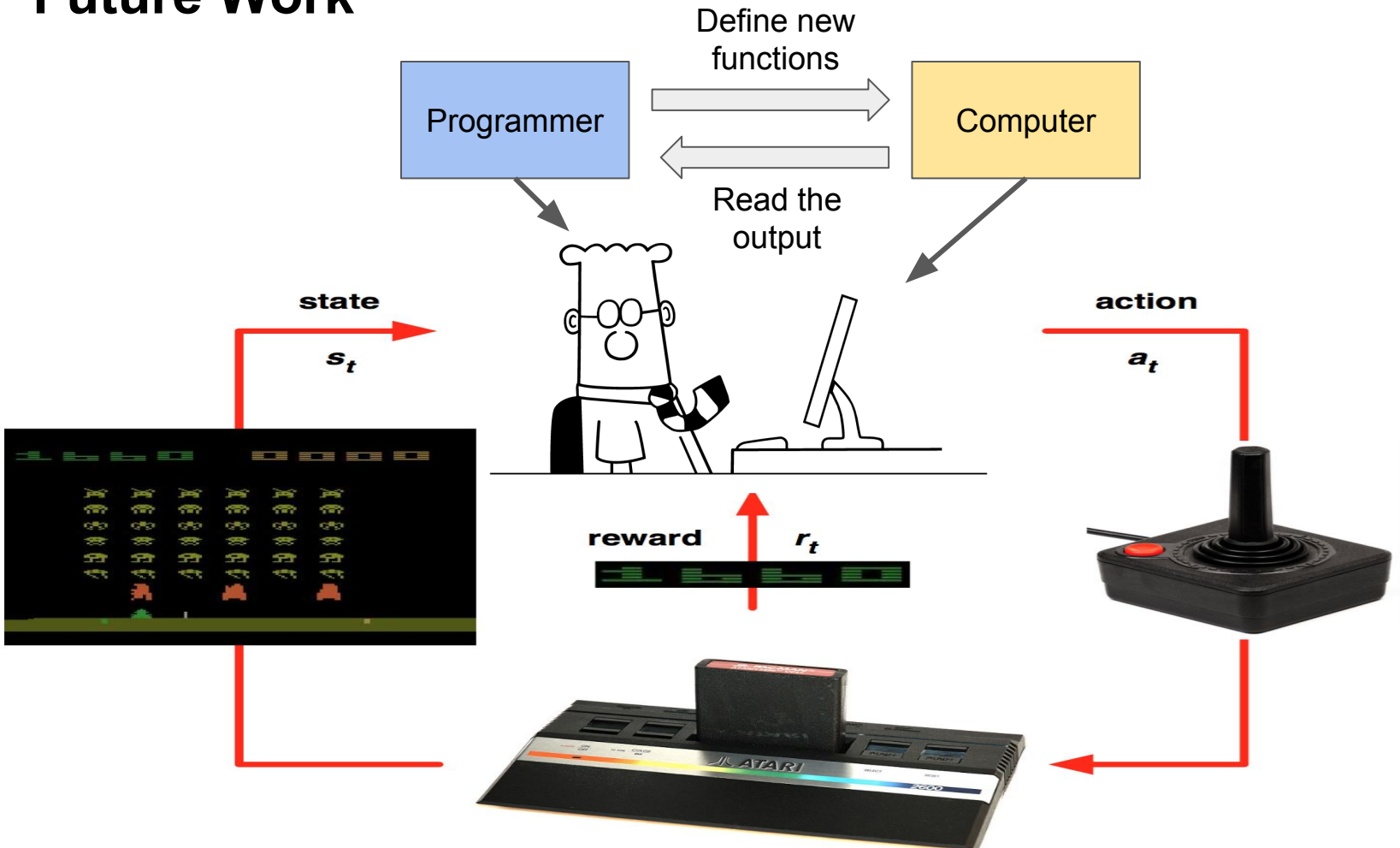


Policy gradient

Mix in approximate gold programs to bootstrap and stabilize training



Future Work


Define new functions
Programmer
Computer
Read the output
state
s_t
action
a_t
reward
r_t
ATARI

More Ablation Analysis

- **Curriculum Learning**
  - Gradually increasing the program complexity during IML training

Settings	Avg. Prec.@Best	Avg. Rec.@Best	Avg. F1@Best	Acc.@Best
No curriculum	79.1	91.1	78.5	67.2
Curriculum	88.6	96.1	89.5	79.8

- **Reduce overfitting**

Settings	Δ Avg. F1@1
-Pretrained word embeddings	-5.5
-Pretrained relation embeddings	-2.7
-Dropout on GRU input and output	-2.4
-Dropout on softmax	-1.1
-Anonymize entity tokens	-2.0

Inspired by STAGG [Yih, et al 2016]

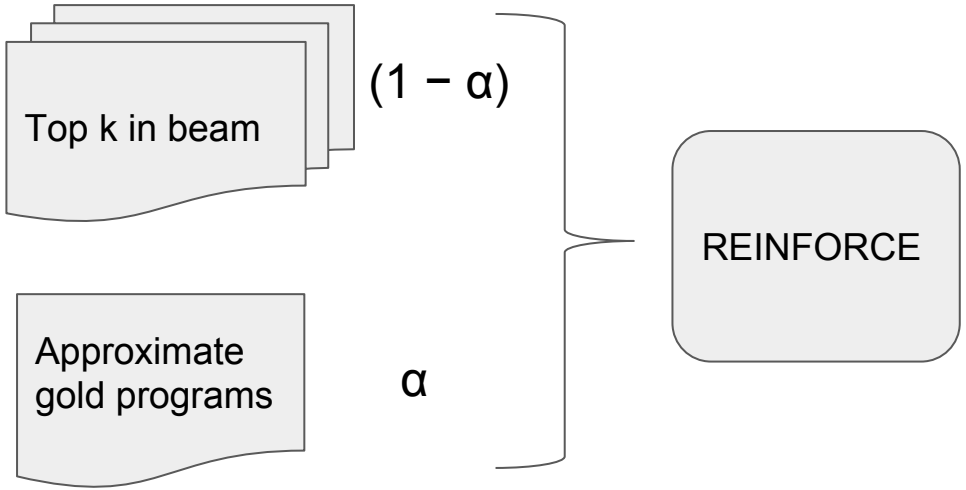
Curriculum Learning

- Gradually increasing the program complexity during ML training
  - First run iterative ML training with only the "Hop" function and the maximum number of expressions is 2
  - Then run iterative ML training again with all functions, and the maximum number of expressions is 3. The relations used by the "Hop" function are restricted to those that appeared in the best programs from in first one
- A lot of search failures without curriculum learning

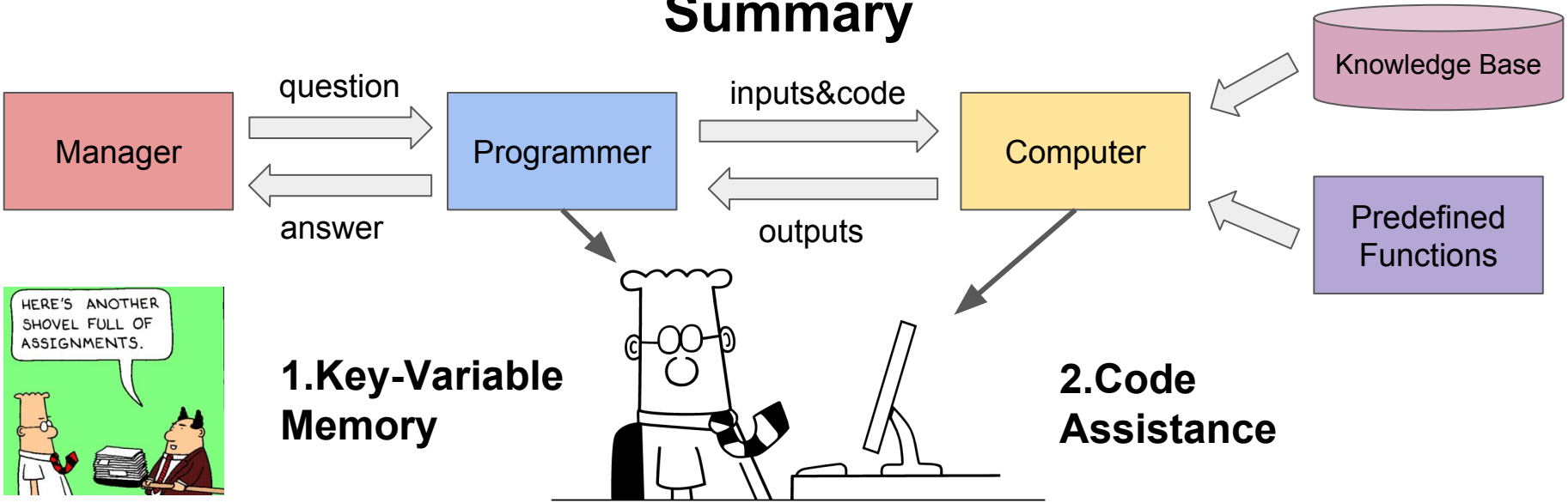
Settings	Avg. Prec.@Best	Avg. Rec.@Best	Avg. F1@Best	Acc.@Best
No curriculum	79.1	91.1	78.5	67.2
Curriculum	88.6	96.1	89.5	79.8

Augmented REINFORCE

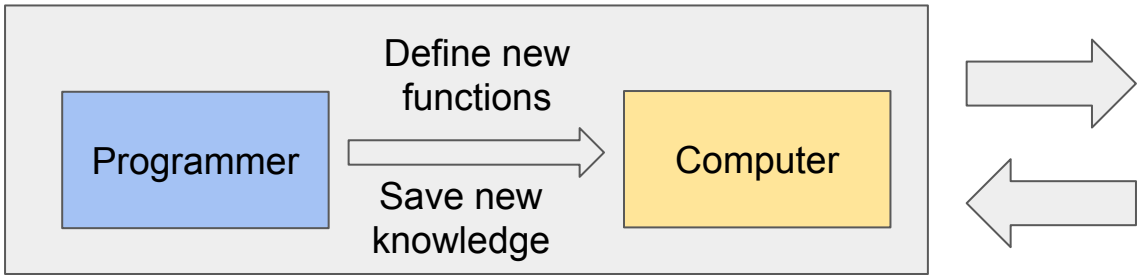
- Can’t do supervised learning, because only weak supervision available…
- shi



Summary

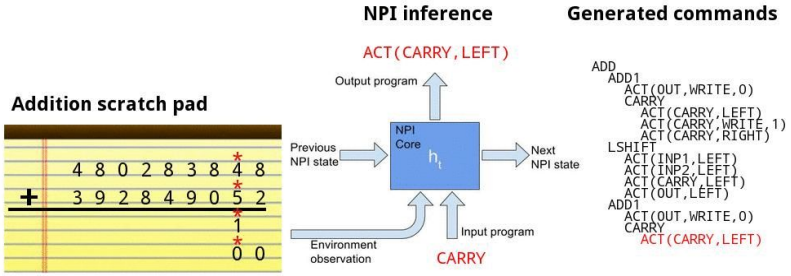


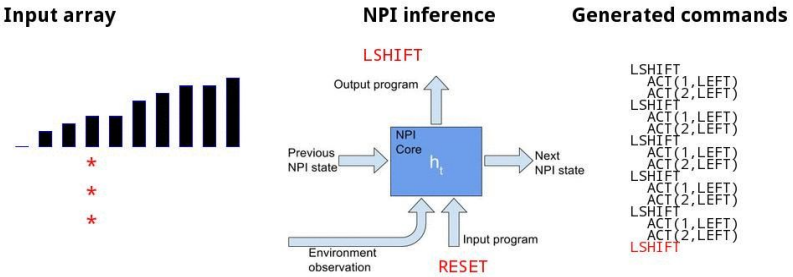
Future Work



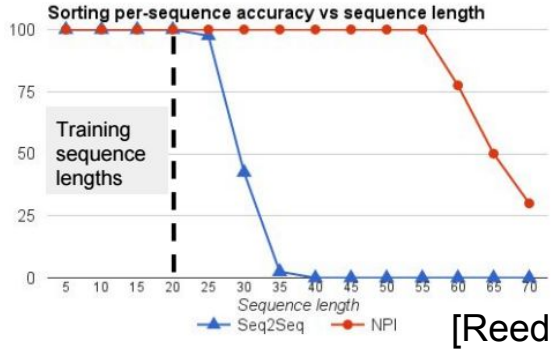
Why not give NN a real programming language?

- Impressive example to show NN can learn addition and sorting, but...



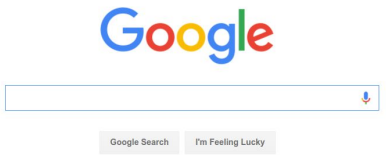


- The operations learned are not as scalable and precise.



[Reed & Freitas 2015]

- Why not leverage existing modules which are scalable and precise?



[Zaremba & Sutskever 2016]

Overview

- Semantic parsing: (updated) WebQuestions dataset
- Neural program induction
- Manager-Programmer-Computer (MPC) framework
- Neural Symbolic Machine
- Experiments and analysis

Inspired by STAGG [Yih, et al 2016]

Curriculum Learning

- Gradually increasing the program complexity during ML training
  - First run iterative ML training with only the "Hop" function and the maximum number of expressions is 2
  - Then run iterative ML training again with all functions, and the maximum number of expressions is 3. The relations used by the "Hop" function are restricted to those that appeared in the best programs from in first one
- A lot of search failures without curriculum learning

Settings	Avg. Prec.@Best	Avg. Rec.@Best	Avg. F1@Best	Acc.@Best
No curriculum	79.1	91.1	78.5	67.2
Curriculum	88.6	96.1	89.5	79.8

Reduce Overfitting

- With all these techniques the model is still overfitting
  - Training F1@1 = 83.0%
  - Validation F1@1 = 67.2%

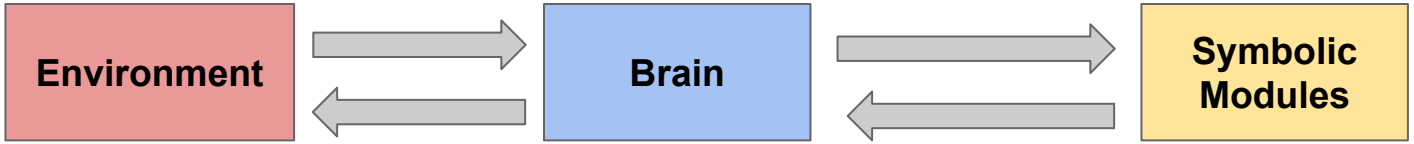
Settings	Δ Avg. F1@1
-Pretrained word embeddings	-5.5
-Pretrained relation embeddings	-2.7
-Dropout on GRU input and output	-2.4
-Dropout on softmax	-1.1
-Anonymize entity tokens	-2.0

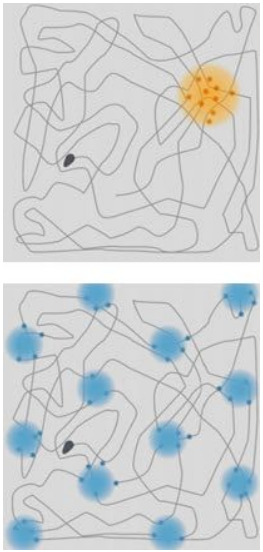
Overview

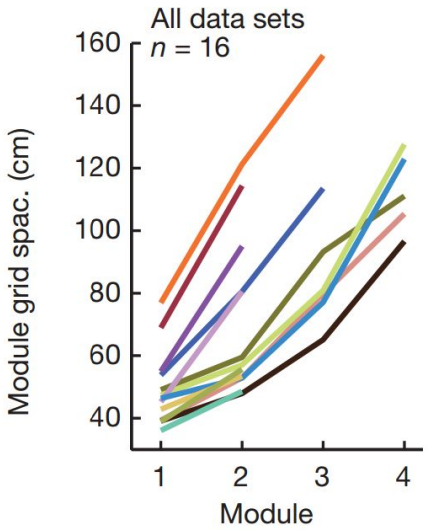
- Semantic parsing: (updated) WebQuestions dataset
- Neural program induction
- Manager-Programmer-Computer (MPC) framework
- Neural Symbolic Machine
- Experiments and analysis

[Stensola+ 2012]

Symbolic Machines in Brains







Mean grid spacing for all modules (M1–M4) in all animals (colour-coded)

- 2014 Nobel Prize in Physiology or Medicine awarded for ‘inner GPS’ research

- Positions are represented as discrete numbers in animals' brains, which enable accurate and autonomous calculations

Overview

- Semantic parsing: (updated) WebQuestions dataset
- Neural program induction
- Manager-Programmer-Computer (MPC) framework
- Neural Symbolic Machine
- Experiments and analysis

Knowledge Base & Semantic Parsing

- Knowledge graph
  - Let E denote a set of entities (e.g., ABELINCOLN), and
  - Let P denote a set of relations (or properties, e.g., PLACEOFBIRTH)
  - A knowledge base K is a set of assertions or triples $(e1, p, e2) \in E \times P \times E$
    e.g., (ABELINCOLN, PLACEOFBIRTH, HODGENVILLE)

- Semantic parsing
  - Given a knowledge base K, and a question $q = (w1 , w2 , ..., wk)$,
  - Produce a program or logical form z that
    when executed against K generates the right answer y

Lisp: High-level Language with Uniform Syntax

- Predefined functions, equivalent to a subset of λ-calculus
  - A program C is a list of expressions (c1...cl)
  - An expression is either a special token "Return" or a list "(F A0 ... Ak)"
  - F is one of the functions

$$
\begin{array}{c}
(\ \textit{Hop}\ v\ p\ ) \Rightarrow \{e_2 | e_1 \in v, (e_1, p, e_2) \in \mathbb{K}\} \\
(\ \textit{ArgMax}\ v\ p\ ) \Rightarrow \{e_1 | e_1 \in v, \exists e_2 \in \mathcal{E} : (e_1, p, e_2) \in \mathbb{K}, \forall e : (e_1, p, e) \in \mathbb{K}, e_2 \geq e\} \\
(\ \textit{ArgMin}\ v\ p\ ) \Rightarrow \{e_1 | e_1 \in v, \exists e_2 \in \mathcal{E} : (e_1, p, e_2) \in \mathbb{K}, \forall e : (e_1, p, e) \in \mathbb{K}, e_2 \leq e\} \\
(\ \textit{Equal}\ v_1\ v_2\ p\ ) \Rightarrow \{e_1 | e_1 \in v_1, \exists e_2 \in v_2 : (e_1, p, e_2) \in \mathbb{K}\}
\end{array}
$$

  - An argument Ai can be either a relation $p \in P$ or a variable v
  - A variable v is a special token (e.g. "R1") representing a list of entities

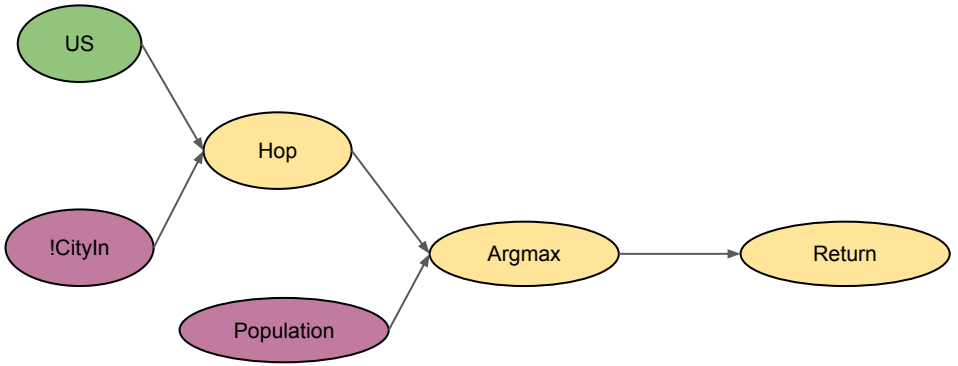
Program as a sequence of tokens

Question: Largest city in US

Seq2Seq

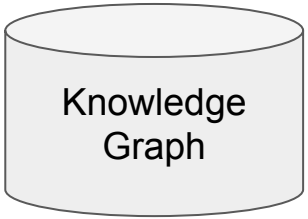
```
(define v0 US)
(define v1 (Hop v0 ?CityIn))
(define v2 (Argmax v1 Population))
(return v2)
```

Lisp Code



Output: NYC

Entities	Relations	Functions
US	CityInCountry	Hop
Obama	BeersFrom	ArgMax
.....	.....	.....



Key Challenges

- Language mismatch
  - Lots of ways to ask the same question
    “What was the date that Minnesota became a state?”
    “When was the state Minnesota created?”
  - Need to map them to the predicate defined in KB
    location.dated_location.date_founded
- Compositionality
  - The semantics of a question may involve multiple predicates and entities
- Large search space
  - Some Freebase entities have >160,000 immediate neighbors
  - 26k predicates in Freebase

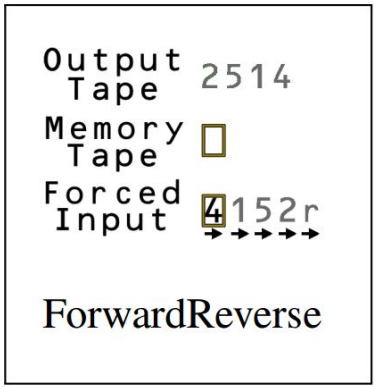
Slides from [Yih+ 2016]

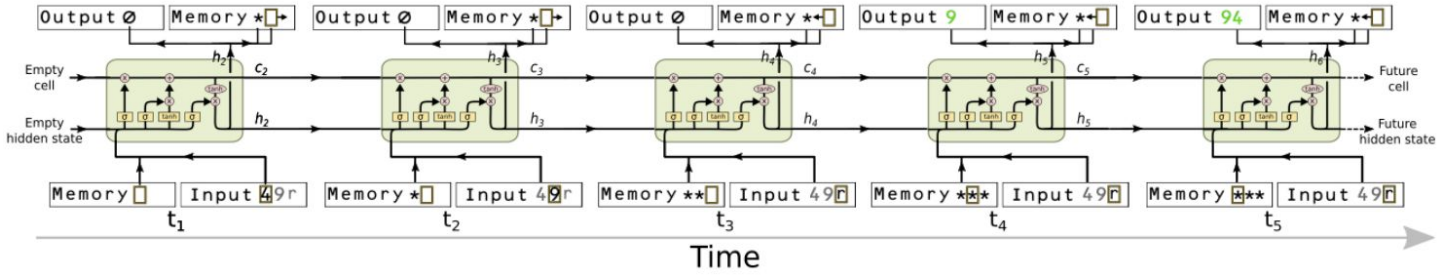
[Zaremba&Sutskever 2016]

Reinforcement Learning Neural Turing Machines

- Interact with a discrete Interfaces
  - a memory Tape, an input Tape, and an output Tape
- Use Reinforcement Learning algorithm to train
- Solve simple algorithmic tasks
  - E.g., reversing a string

Need higher level programming language for semantic parsing





Key-Variable Memory

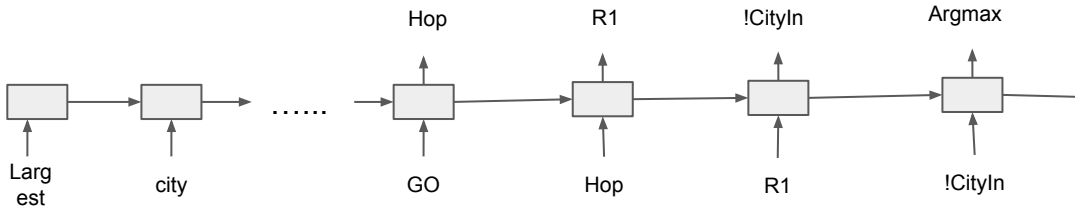
- The memory is 'symbolic'
  - Variables are symbols referencing intermediate results in computer
  - No need to have embeddings for hundreds of millions of entities in KG
  - Keys are differentiable, but variables are not

- Human use names/comments to index intermediate results

Comments	Variable
Entity extracted from the word after “city in”	R1(m.USA)
Generated by querying v1 with !CityIn	R2(a list of US cities)

- NN use embeddings (outputs of GRUs) to index results

Embeddings	Variable
[0.1, -0.2, 0.3, …]	R1(m.USA)
[0.8, 0.5, -0.3, …]	R2(a list of US cities)



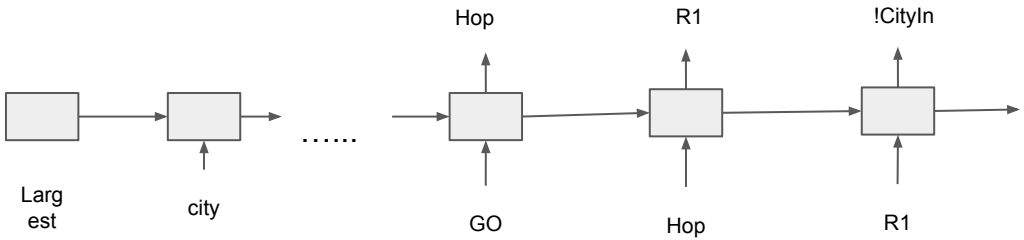
Neural Computer Interface

- A Strong IDE / Interpreter helps reduce the search space
  - Exclude the invalid choices that will cause syntax and semantic error

Syntax check:
Only a variable can follow ‘Hop’

Semantic check:
only relations that are connected to R1 can be used **~ 20k => ~ 100**

Implemented as a changing mask on decoding vocabulary



Non-differentiable => REINFORCE Training

- Optimizing expected F1

$$J^{RL}(\theta) = \sum_q \mathbb{E}_{P(a_{0:T}|q,\theta)}[R(q, a_{0:T})]$$

- Use baseline B(q) to reduces variance without changing the optima

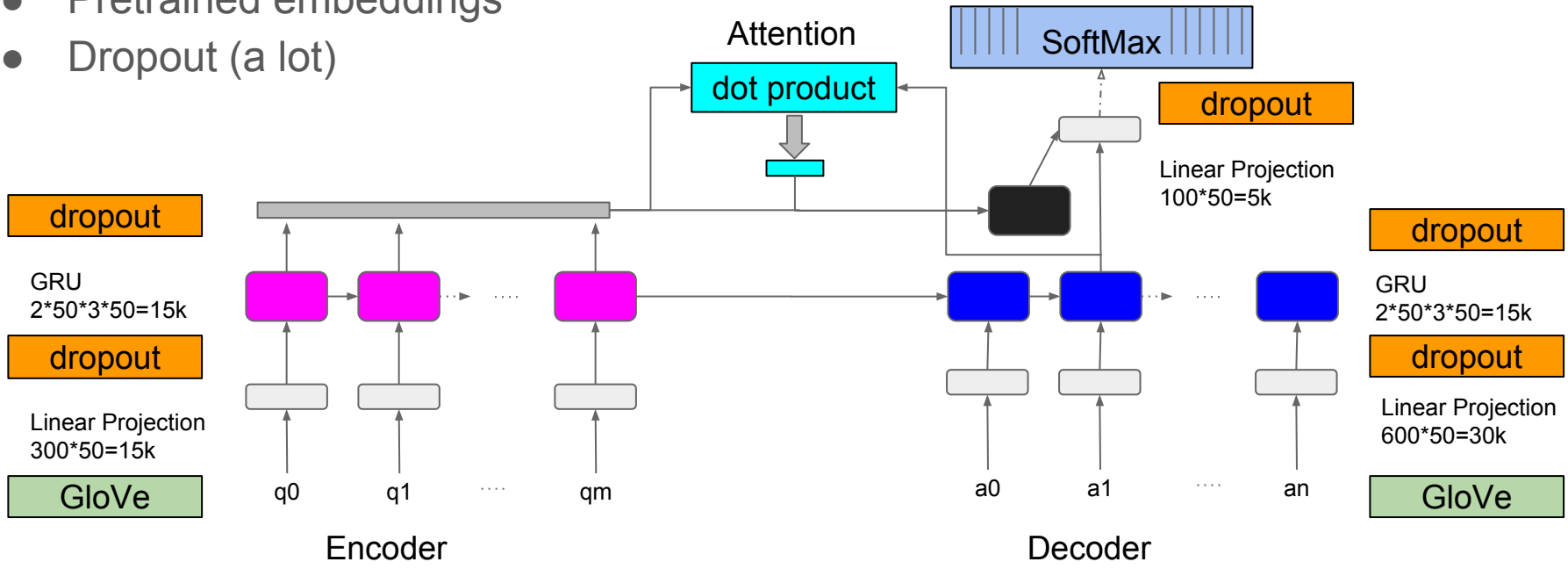
$$\nabla_\theta J^{RL}(\theta) = \sum_q \sum_{a_{0:T}} P(a_{0:T}|q,\theta)[R(q, a_{0:T}) - B(q)]\nabla_\theta \log P(a_{0:T}|q,\theta)$$

$$B(q) = \sum_{a_{0:T}} P(a_{0:T}|q,\theta)R(q, a_{0:T})$$

- Gradient computation is approximated by beam search instead of sampling

Model Architecture

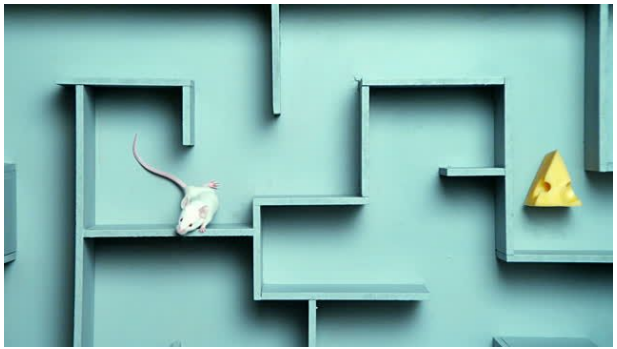
- Small model: 15k+30k+15k*2+5k = 80k params
- Dot product attention
- Pretrained embeddings
- Dropout (a lot)



Sampling v.s. Beam search

- Decoding uses beam search
  - Use top k in beam (normalized probabilities) to compute gradients
  - Reduce variance and estimate the baseline better
- The coding environment is deterministic. Closer to a maze than Atari game.

Stochastic



Deterministic

Problem with REINFORCE

Training is slow and get stuck on local optimum

- Large search space
  - model probability of good programs with non-zero F1 is very small
- Large beam size
  - Normalized probability small
  - Decoding and training is slow because larger number of sequences
- Small beam size
  - Good programs might fall off the beam

Solution:
Add some gold programs into the beam with reasonably large probability... but we **don't** have gold programs, only **weak supervision**

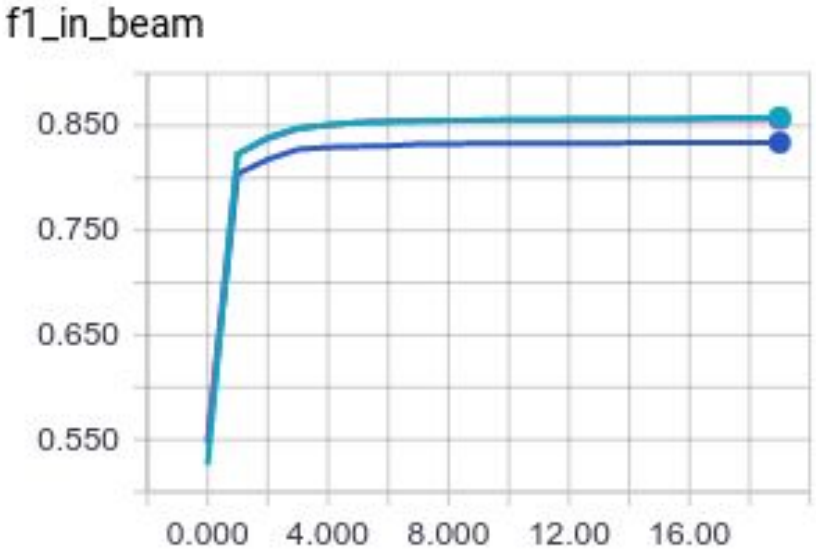
Finding Approximate Gold Programs

- Ideally we want to do supervised pretraining for REINFORCE, but we only have weak supervision
- Use an iterative process interleaving decoding with large beam and maximum likelihood training

- Training objective:

$$J^{ML}(\theta) = \sum_{q} \log P(a^{best}_{0:T}(q)|q,\theta)$$

- Training is fast and has a bootstrap effect



Drawbacks of the ML objective

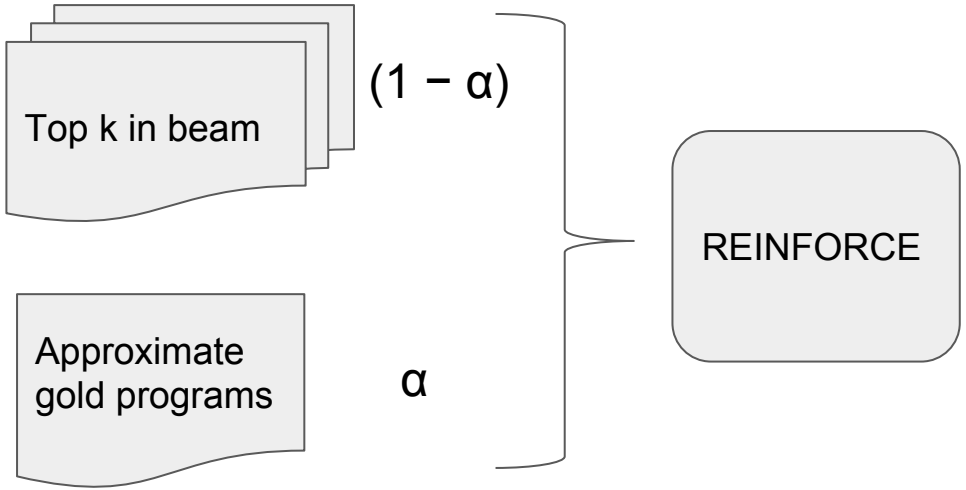
- Not directly optimizing expected F1

- The best program for a question could be a spurious program that accidentally produced the correct answer, and thus does not generalize to other questions
  - e.g., answering PLACEOFBIRTH with PLACEOFDEATH

- Because training lacks explicit negative examples, the model fails to distinguish between tokens that are related to one another
  - e.g., PARENTSOF vs. SIBLINGSOF vs. CHILDRENOF

Augmented REINFORCE

- Add the approximate gold program into the final beam with probability α, and the probabilities of the original programs in the beam are normalized to be (1 − α).
- The rest of the process is the same as in standard REINFORCE



Algorithm

- **MLE** for fast training
- **Beam search** for better exploration
- **REINFORCE** for optimizing the correct objective
- **Experience replay** to improve training stability

Input: question-answer pairs $\mathbb{D} = \{(x_i, y_i)\}$, mix ratio α, reward function $R(\cdot)$, training iterations N_{ML}, N_{RL}, and beam sizes B_{ML}, B_{RL}.
Procedure:
Initialize $C_x^* = \emptyset$ the best program so far for x
Initialize model θ randomly ▷ Iterative ML
for $n = 1$ to N_{ML} **do**
  for (x, y) in D **do**
    $\mathbb{C} \leftarrow$ Decode B_{ML} programs given x
    for j in $1...|\mathbb{C}|$ **do**
      if $R_{x,y}(C_j) > R_{x,y}(C_x^*)$ **then** $C_x^* \leftarrow C_j$
  $\theta \leftarrow$ ML training with $\mathbb{D}_{ML} = \{(x, C_x^*)\}$
Initialize model θ randomly ▷ REINFORCE
for $n = 1$ to N_{RL} **do**
  $\mathbb{D}_{RL} \leftarrow \emptyset$ is the RL training set
  for (x, y) in D **do**
    $\mathbb{C} \leftarrow$ Decode B_{RL} programs from x
    for j in $1...|\mathbb{C}|$ **do**
      if $R_{x,y}(C_j) > R_{x,y}(C_x^*)$ **then** $C_x^* \leftarrow C_j$
    $\mathbb{C} \leftarrow \mathbb{C} \cup \{C_x^*\}$
    for j in $1...|\mathbb{C}|$ **do**
      $\hat{p}_j \leftarrow (1-\alpha) \cdot \frac{p_j}{\sum_{j'} p_{j'}}$ where $p_j = P_\theta(C_j \mid x)$
      if $C_j = C_x^*$ **then** $\hat{p}_j \leftarrow \hat{p}_j + \alpha$
      $\mathbb{D}_{RL} \leftarrow \mathbb{D}_{RL} \cup \{(x, C_j, \hat{p}_j)\}$
  $\theta \leftarrow$ REINFORCE training with $\mathbb{D}_{RL}$

Overview

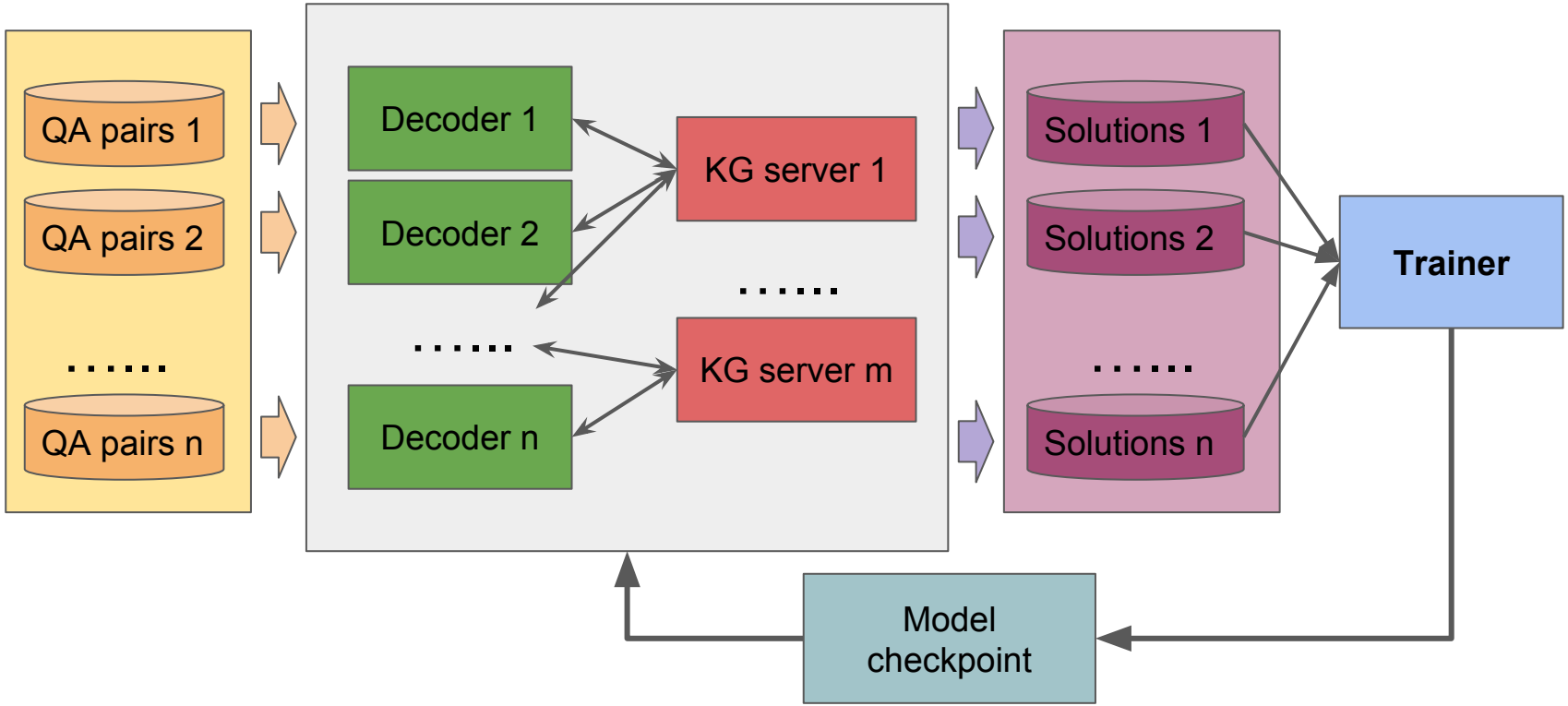
- Semantic parsing: (updated) WebQuestions dataset
- Neural program induction
- Manager-Programmer-Computer (MPC) framework
- Neural Symbolic Machine
- Experiments and analysis

Freebase Preprocessing

- Remove predicates which are not related to world knowledge
  - Those starting with "/common/", "/type/", "/freebase/"
- Remove all text valued predicates
  - They are almost never the answer of questions
- Result in a graph which is small enough to fit in memory
  - #Relations=23K
  - #Nodes=82M
  - #Edges=417M

System Architecture

- 200 decoders, 50 KG servers, 1 trainer, 251 machines in total
- The solutions to a query include programs and their rewards



Compare to State-of-the-Art

- First end-to-end neural network to achieve state-of-the-art performance on semantic parsing with weak supervision over large knowledge base
- The performance is approaching state-of-the-art result with full supervision

Model	Avg. Prec.@1	Avg. Rec.@1	Avg. F1@1	Acc.@1
STAGG	67.3	73.1	66.8	58.8
NSM – our model	70.8	76.0	**69.0**	59.5
STAGG (full supervision)	70.9	80.3	71.7	63.9

Augmented REINFORCE

- REINFORCE get stuck at local maxima
- Iterative ML training is not directly optimizing the F1 measure
- Augmented REINFORCE obtains the best performances

Settings	Train Avg. F1@1	Valid Avg. F1@1
iterative ML only	68.6	60.1
REINFORCE only	55.1	47.8
Augmented REINFORCE	83.0	**67.2**

Inspired by STAGG [Yih, et al 2016]

Curriculum Learning

- Gradually increasing the program complexity during ML training
  - First run iterative ML training with only the "Hop" function and the maximum number of expressions is 2
  - Then run iterative ML training again with all functions, and the maximum number of expressions is 3. The relations used by the "Hop" function are restricted to those that appeared in the best programs from in first one

Settings	Avg. Prec.@Best	Avg. Rec.@Best	Avg. F1@Best	Acc.@Best
No curriculum	79.1	91.1	78.5	67.2
Curriculum	88.6	96.1	89.5	79.8

Reduce Overfitting

- With all these techniques the model is still overfitting
  - Training F1@1 = 83.0%
  - Validation F1@1 = 67.2%

Settings	Δ Avg. F1@1
-Pretrained word embeddings	-5.5
-Pretrained relation embeddings	-2.7
-Dropout on GRU input and output	-2.4
-Dropout on softmax	-1.1
-Anonymize entity tokens	-2.0

Example Program

- **Question**: “what college did russell wilson go to?”
- **Generated program**:
  (hop v1 /people/person/education)
  (hop v2 /education/education/institution)
  (filter v3 v0 /common/topic/notable_types)
  <EOP>

  v0 = “College/University” (m.01y2hnl) v1 = “Russell Wilson” (m.05c10yf).

Future work

- Better performance with more training data
- Actions to add knowledge into KG and create new schema
- Language to action

Acknowledgement

- Thanks for discussions and helps from Arvind, Mohammad, Tom, Eugene, Lukasz, Thomas, Yonghui, Zhifeng, Alexandre, John

- Thanks for MSR researchers, who made WebQuestionSP data set available