Neural Symbolic Language Understanding

Ni Lao

11.12.2017

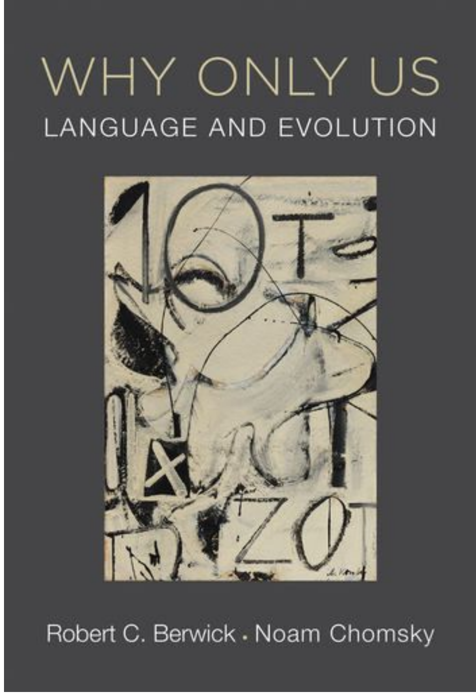
Plan

- **Language & control**
  - *Neural Symbolic Machines*: Semantic Parsing on Freebase with Weak Supervision
    - Chen Liang, Jonathan Berant, Quoc Le, Kenneth Forbus, Ni Lao

- Knowledge & scalability
  - Learning to Organize Knowledge with An *N-Gram Machine*
    - Fan Yang, Jiazhong Nie, William Cohen, Ni Lao

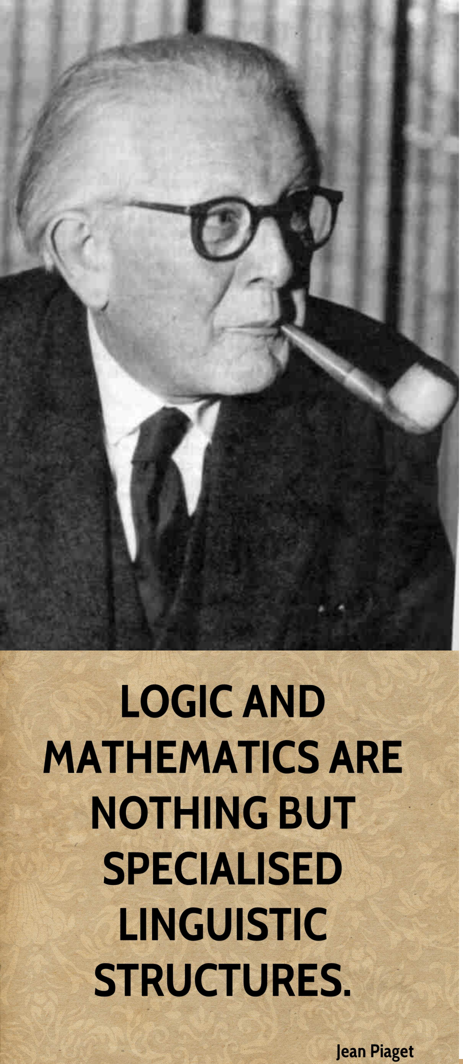
Language & Reasoning

- Language was primarily invented for reasoning
- Communication comes later



ACL [Liang+ 2017]

Language, Translation & Control



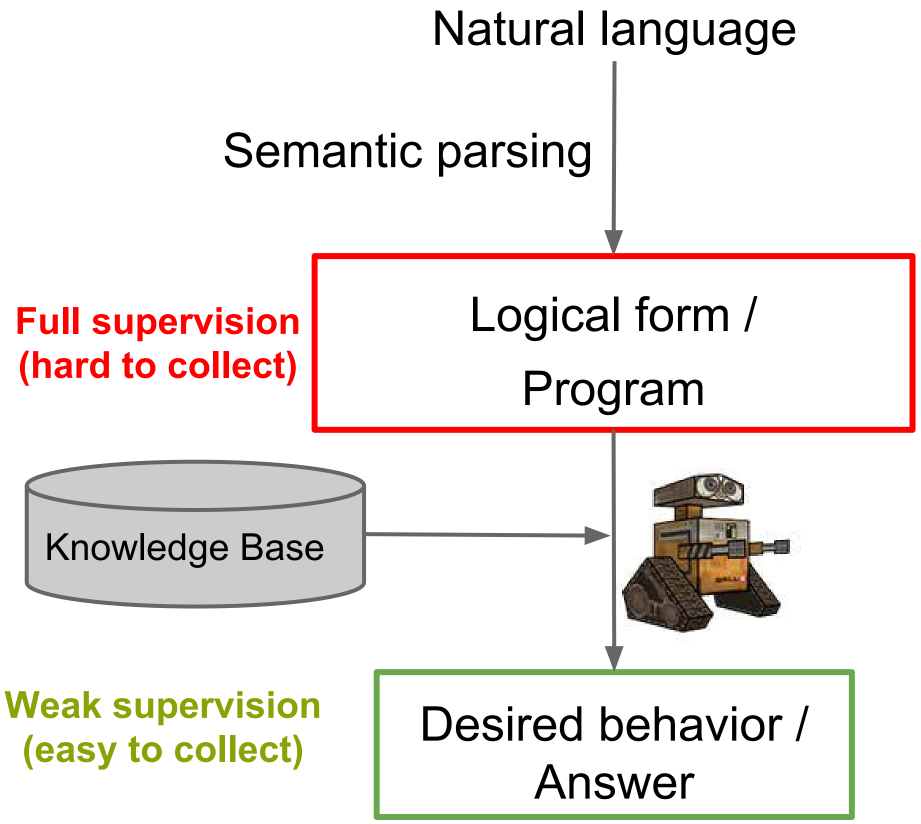
1) Natural languages are programming languages to control human behavior

2) For machines and human to understand each other, they just need translation models trained with control theory

[Berant+ 2013]
[Liang 2013]

Semantic Parsing

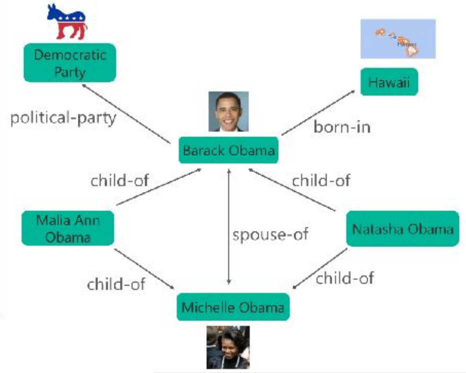
- Question answering with structured data (KG / tables / personal data)
- Voice to action
- Personal assistant
- Entertainment



Question Answering with Knowledge Base

Largest city in US?

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



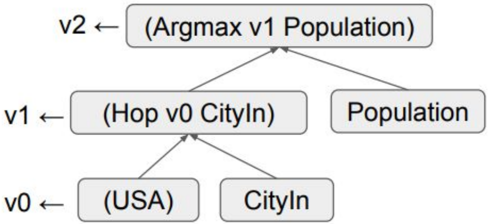
Freebase, DBpedia, YAGO, NELL

NYC

0. Paraphrase

Many ways to ask the same question, e.g., "What was the date that Minnesota became a state?" "When was the state Minnesota created?"

1. Compositionality



2. Large Search Space

E.g., Freebase: 23K predicates, 82M entities, 417M triplets

3. Optimization

Reinforcement Learning is known to be hard given sparse reward

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

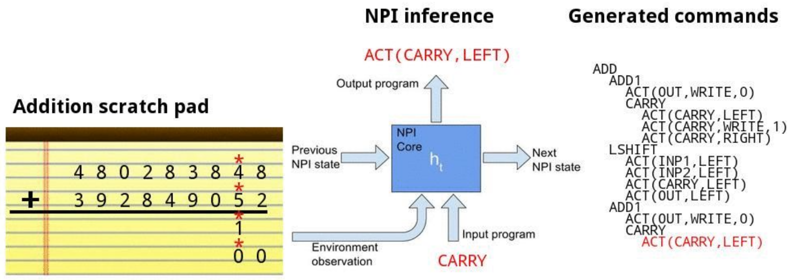
Multiple entities

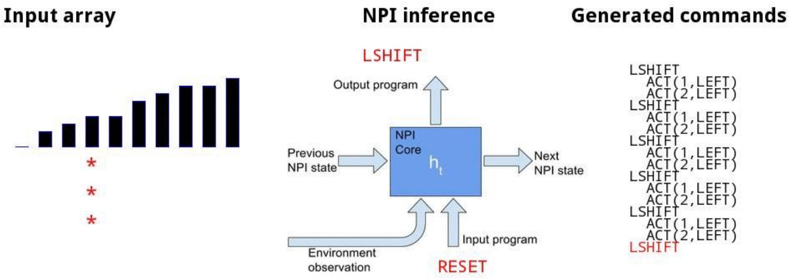
• What do Michelle Obama do for a living?	writer, lawyer
• What character did Natalie Portman play in Star Wars?	Padme Amidala
• What currency do you use in Costa Rica?	Costa Rican colon
• What did Obama study in school?	political science
• What killed Sammy Davis Jr?	throat cancer

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

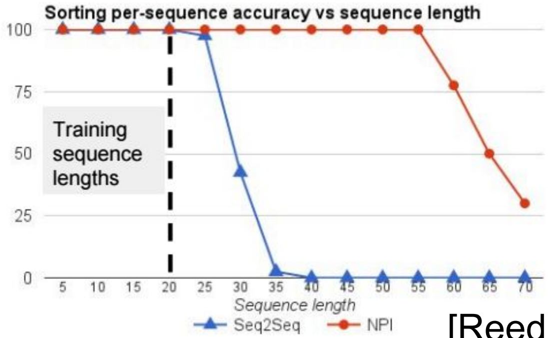
(Scalable) Neural Program Induction

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



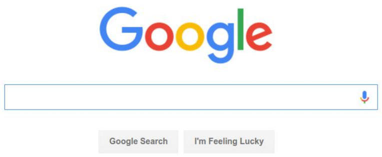


- The learned operations are not as scalable and precise.



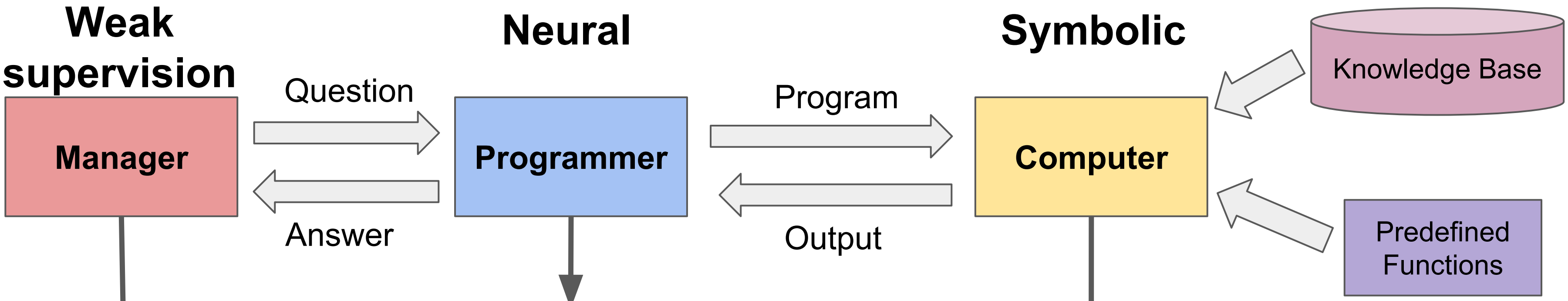
[Reed & Freitas 2015]

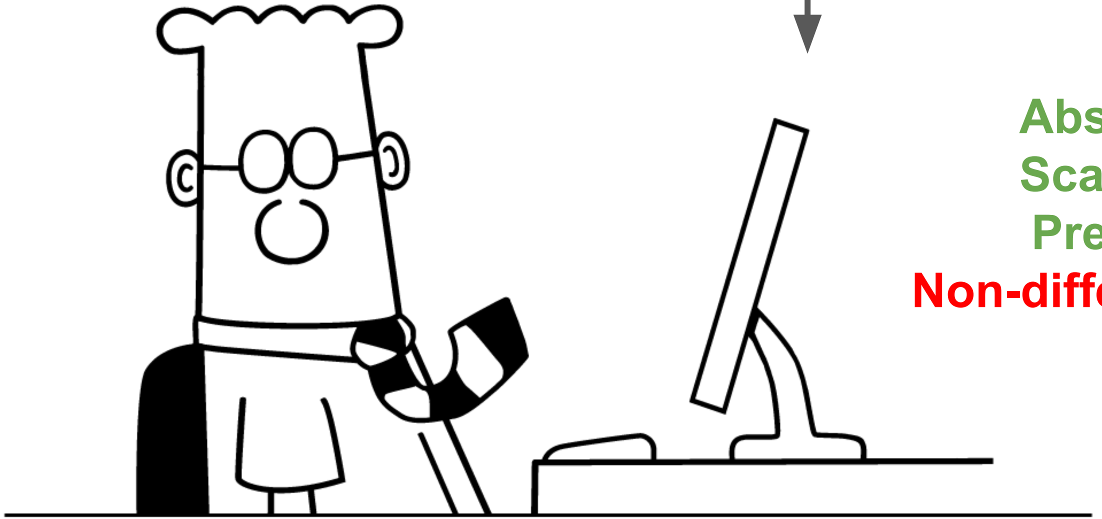
- Why not use existing modules that are scalable, precise and interpretable?



[Zaremba & Sutskever 2016]

Neural Symbolic Machines



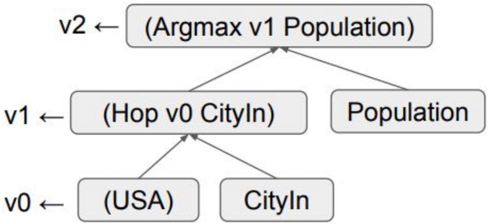


Abstract
Scalable
Precise
Non-differentiable

Contributions

- Simple Seq2Seq model is not enough

1. Compositionality



1. Key-Variable Memory

2. Large Search Space

E.g., Freebase:
23K predicates,
82M entities,
417M triplets

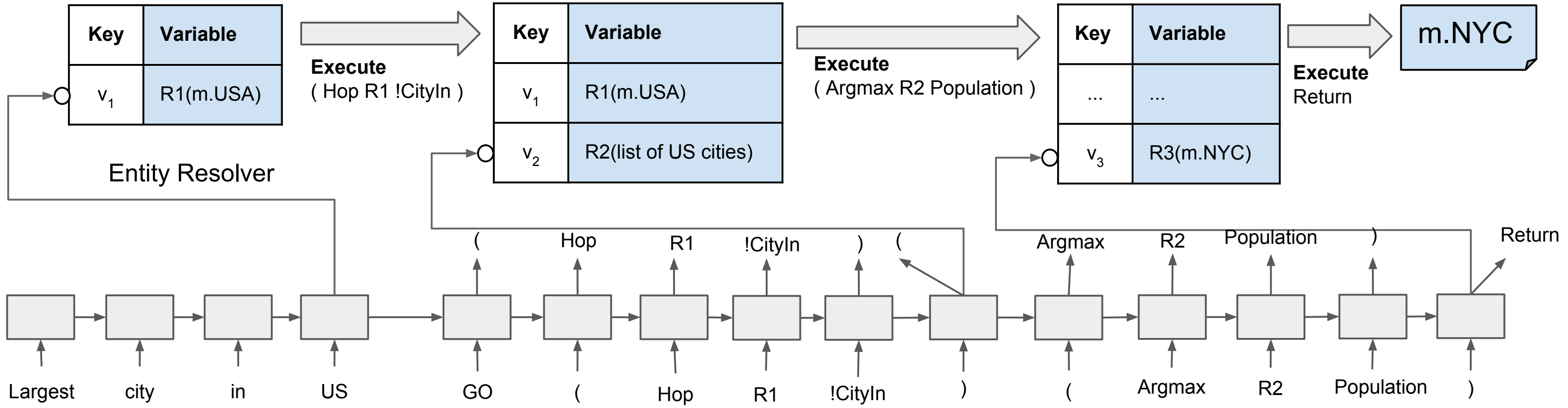
2. Beam search + Code Assistance

3. Optimization

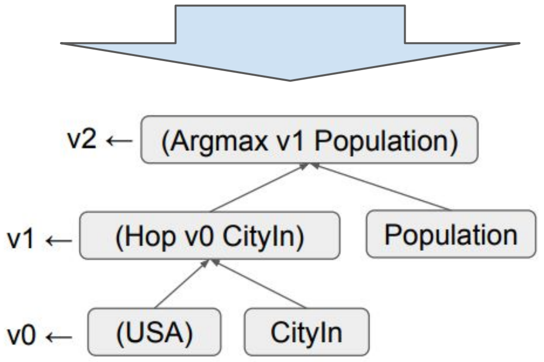
Reinforcement Learning is known to be hard given sparse reward

3. Augmented REINFORCE

Key-Variable Memory for Compositionality

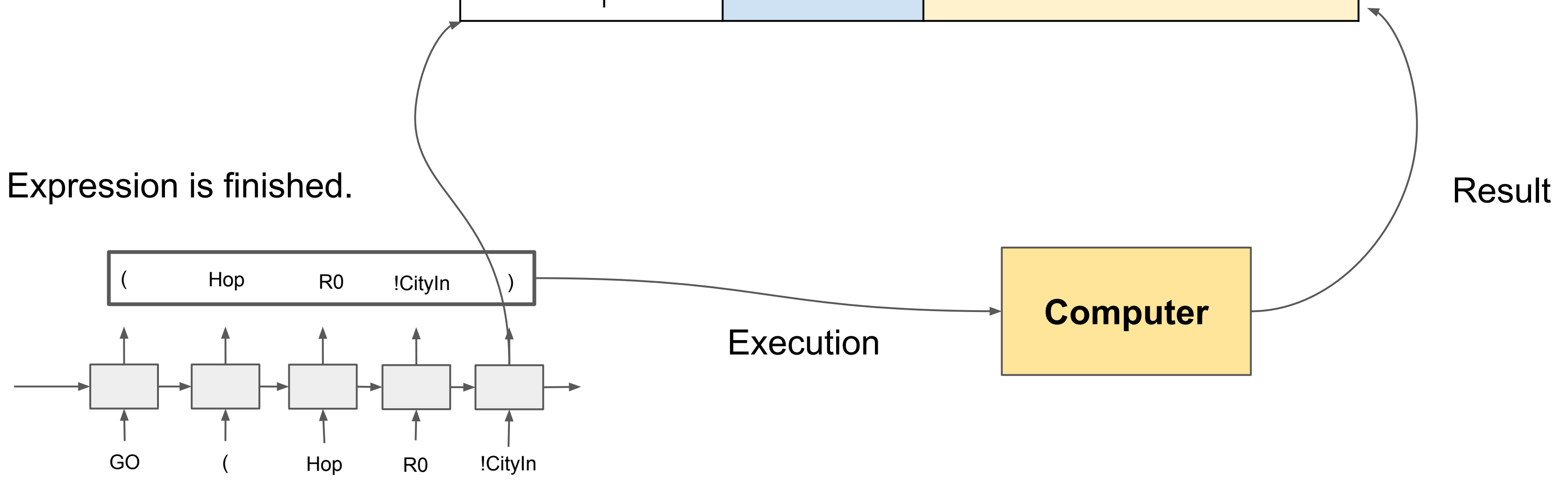


- A linearised bottom-up derivation of the recursive program



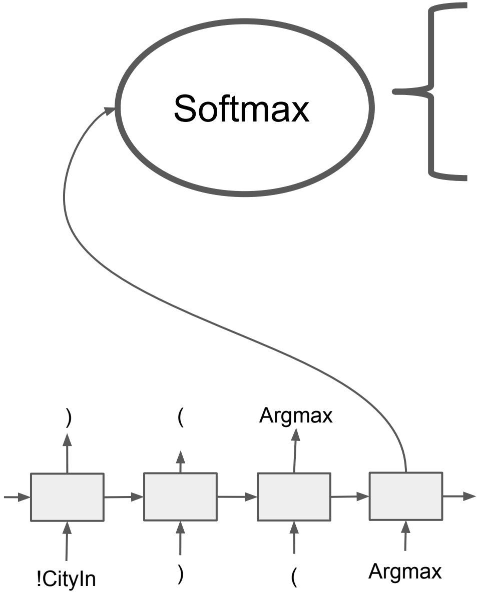
Key-Variable Memory: Save Intermediate Value

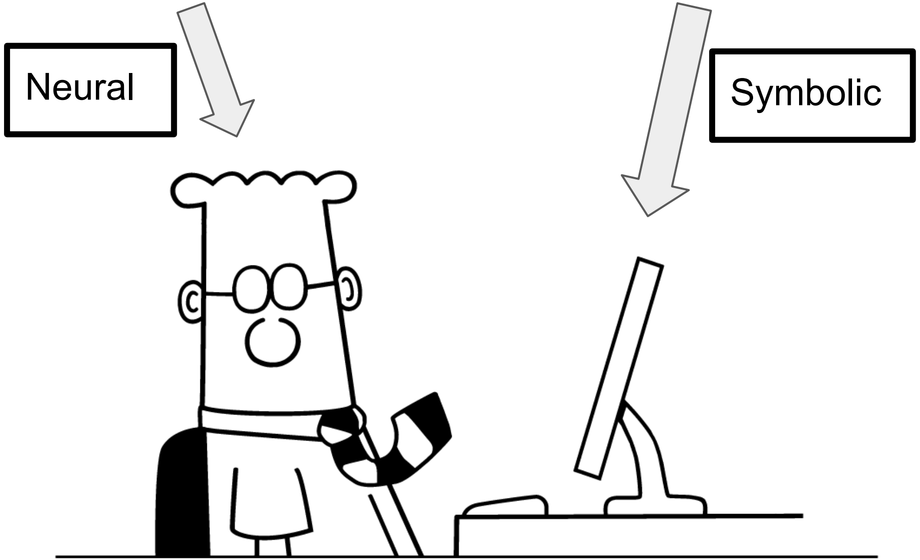
Key (Embedding)	**Variable** (Symbol)	**Value** (Data in Computer)
V_0	R0	m.USA
V_1	R1	[m.SF, m.NYC, ...]



Key-Variable Memory: Reuse Intermediate Value

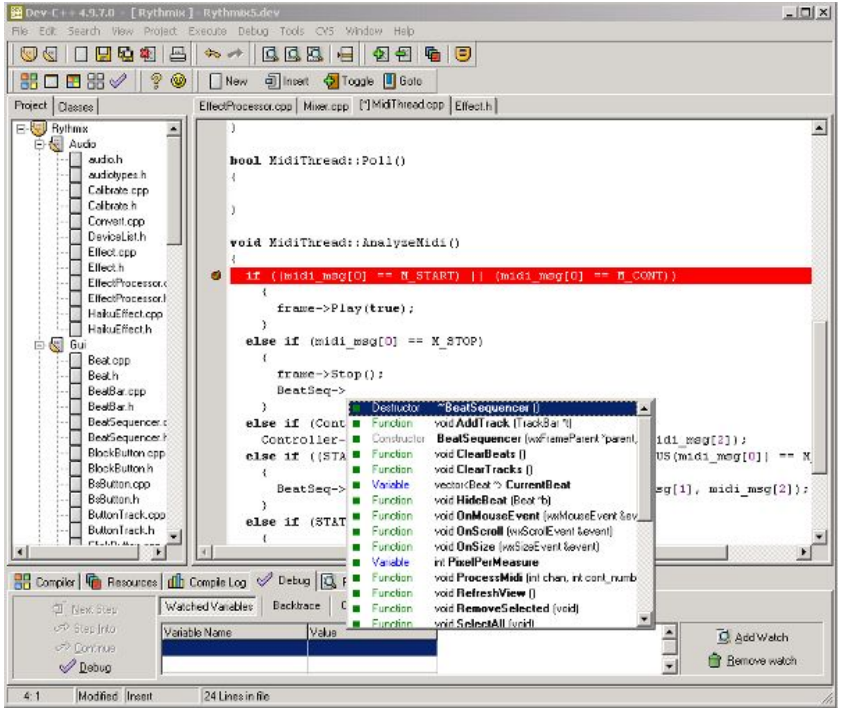
Key (Embedding)	**Variable** (Symbol)	**Value** (Data in Computer)
V_0	`R0`	`m.USA`
V_1	`R1`	`[m.SF, m.NYC, ...]`





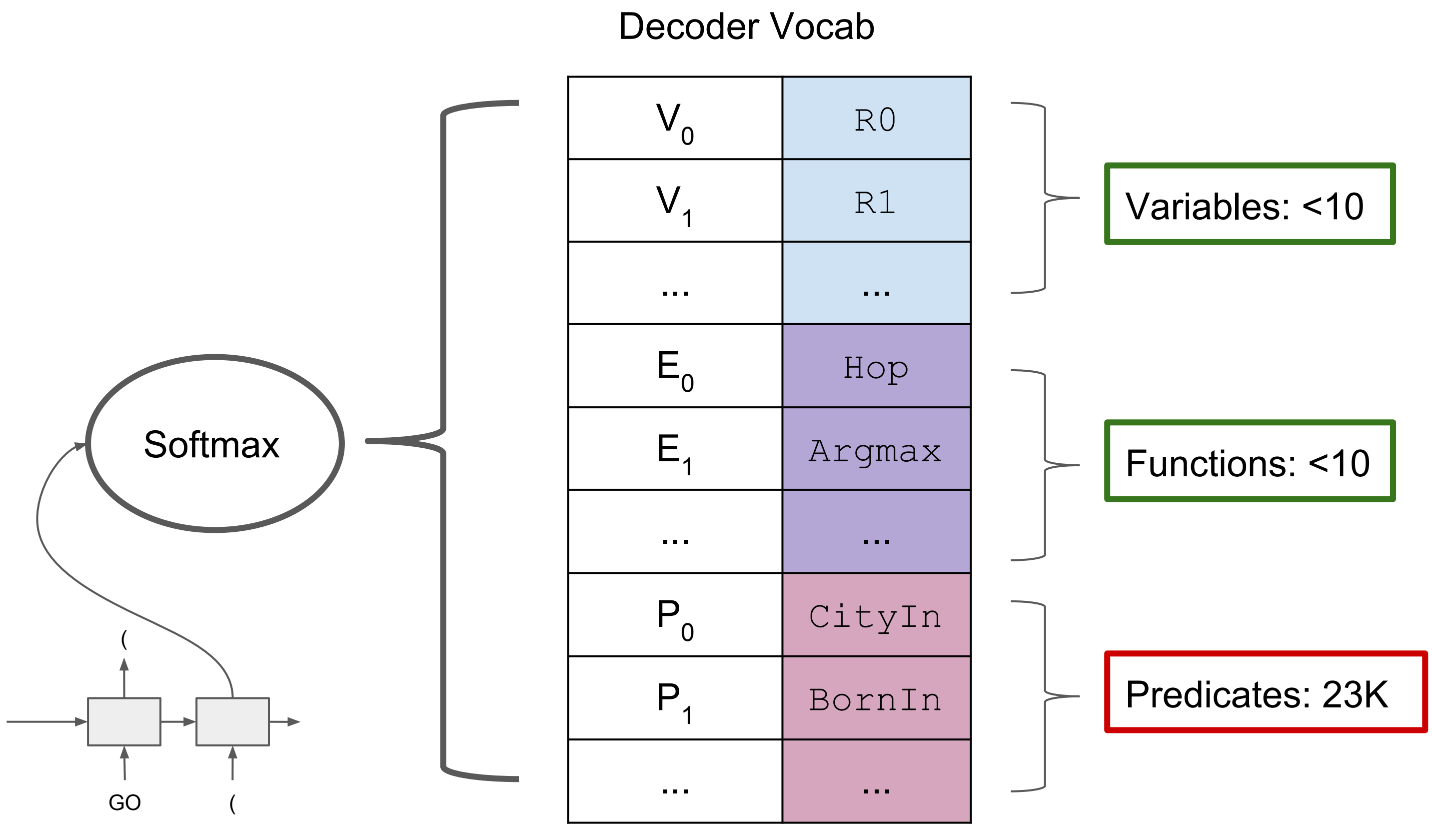
Code Assistance: Prune Search Space

Pen and paper



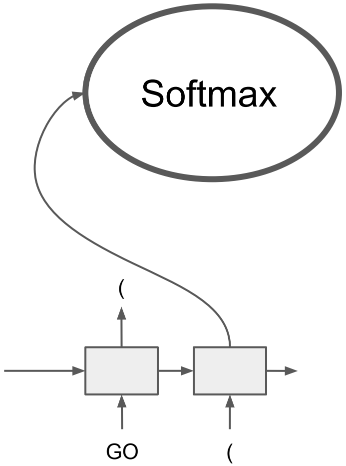
IDE

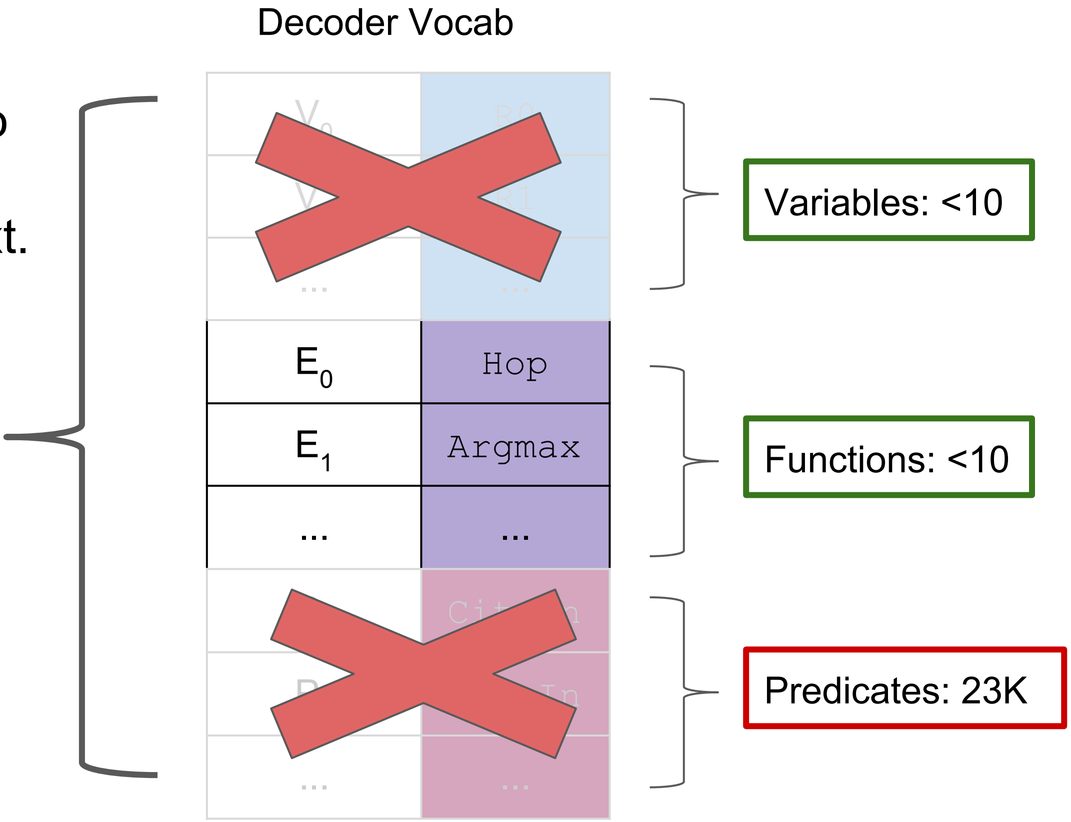
Code Assistance: Syntactic Constraint



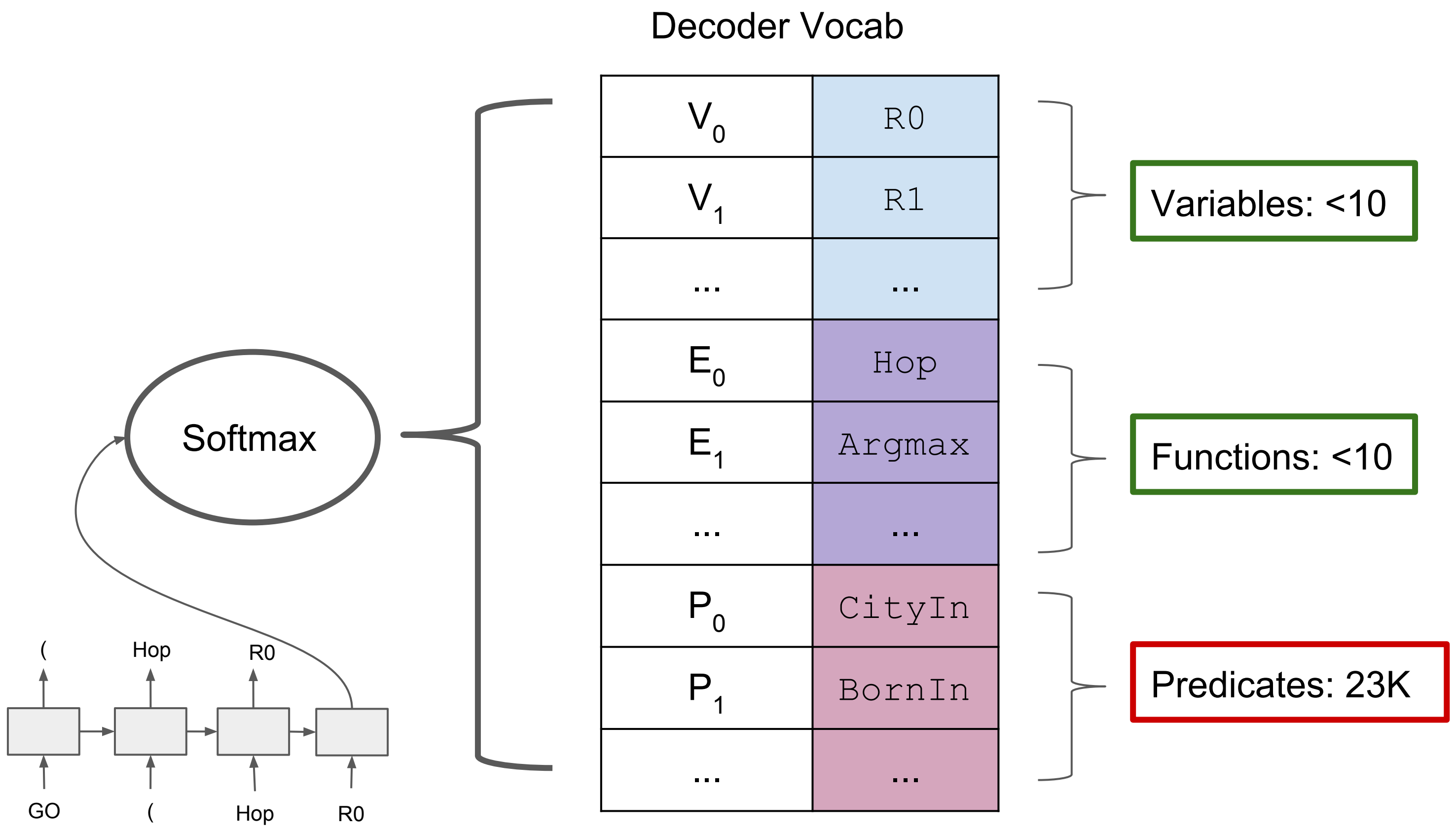
Code Assistance: Syntactic Constraint

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



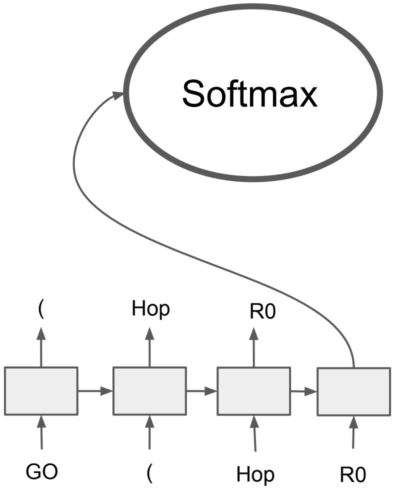


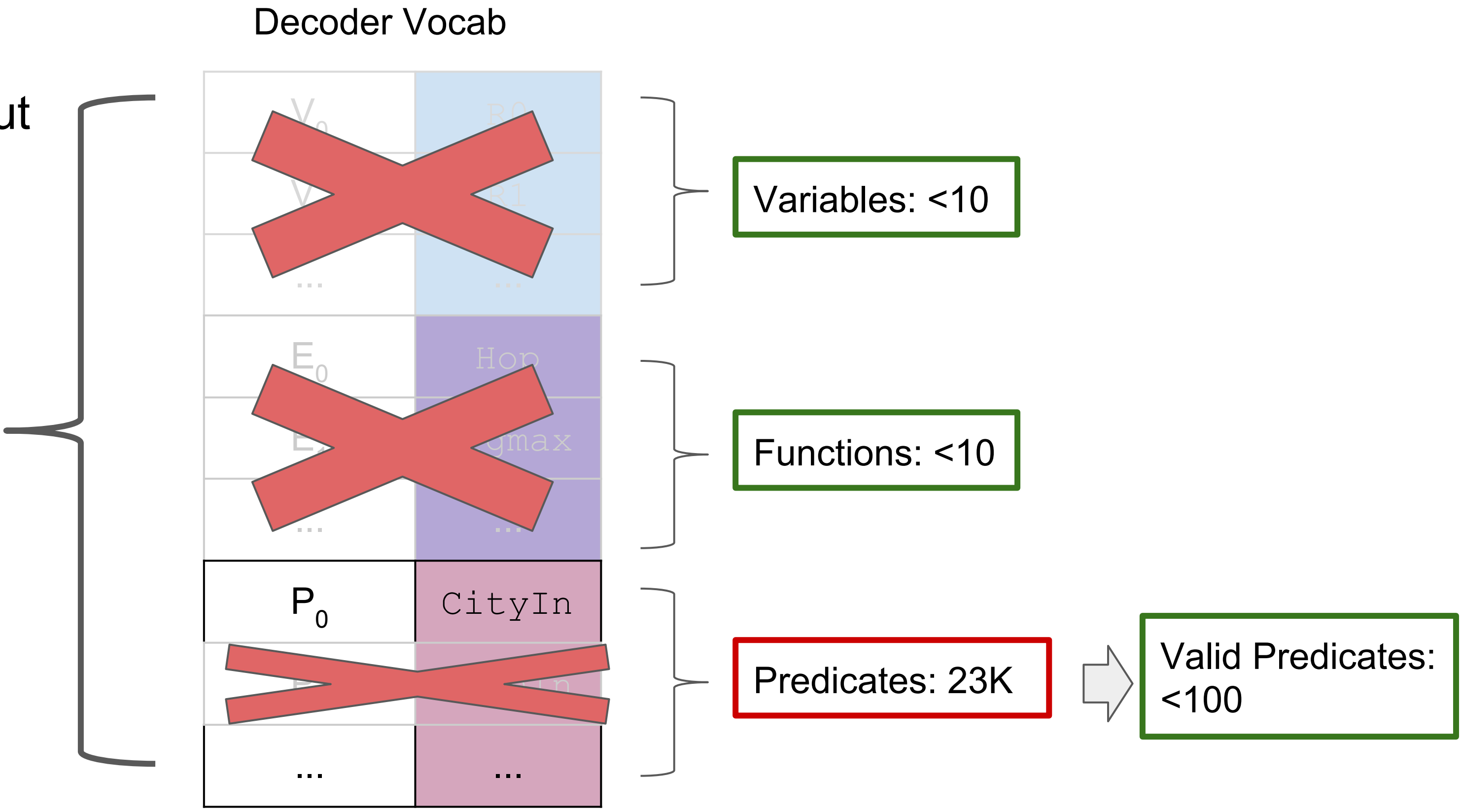
Code Assistance: Semantic Constraint



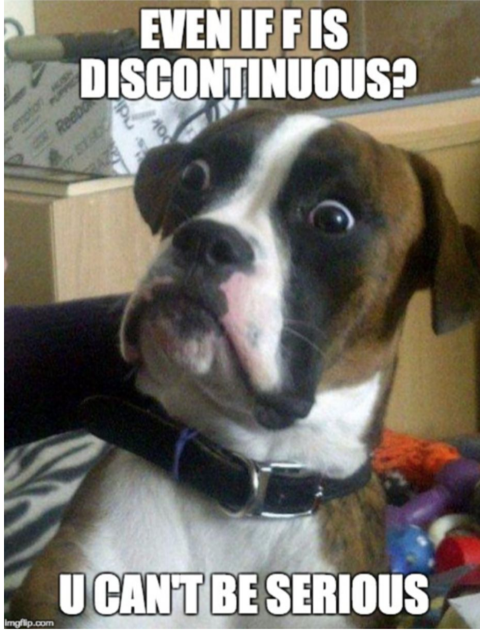
Code Assistance: Semantic Constraint

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





Policy Gradient (REINFORCE)



- MLE optimizes log likelihood of approximate gold programs

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

- RL optimizes the expected reward under a stochastic policy P

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

- The gradient is almost the same as that for MLE except for a weight P(R-B)

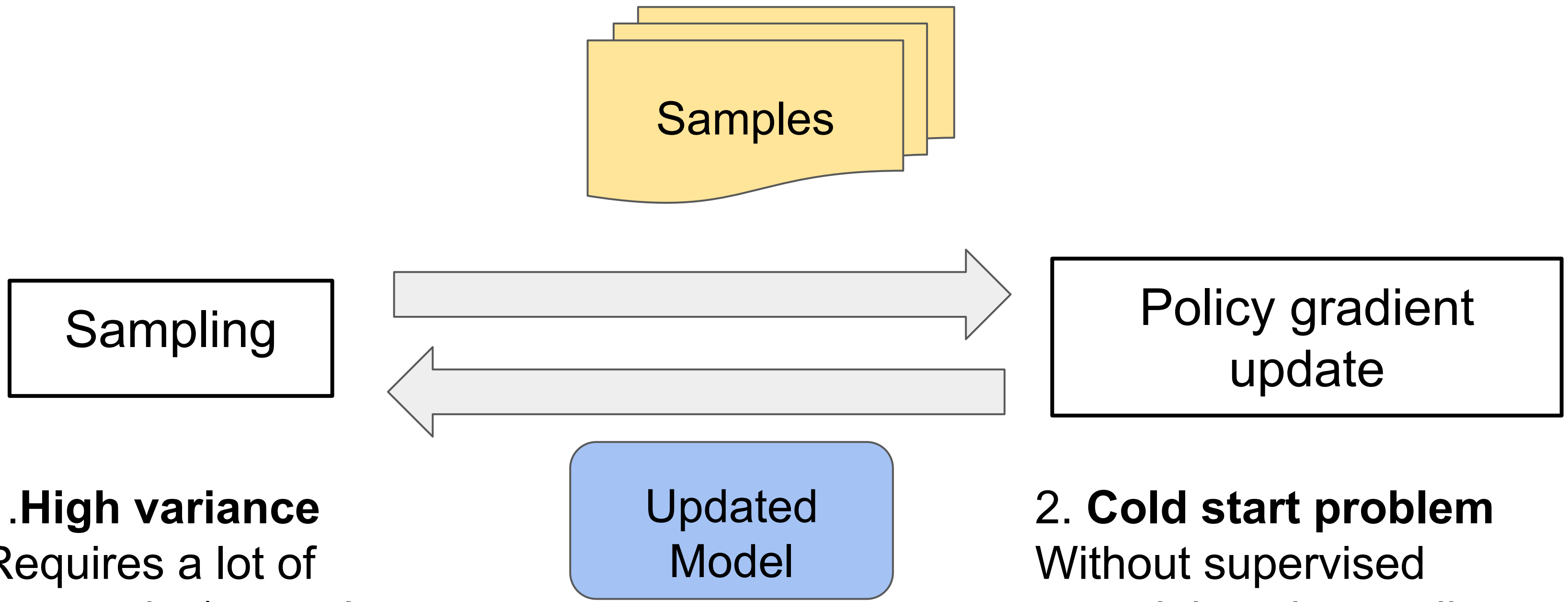
$$\nabla_\theta J^{RL}(\theta) = \sum_q \sum_{a_{0:T}} \boxed{P(a_{0:T}|q, \theta)[R(q, a_{0:T}) - B(q)]} \nabla_\theta \log P(a_{0:T}|q, \theta)$$

- The baseline does not change the solution but improves convergences, e.g.,

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

[Williams 1992]

REINFORCE Training

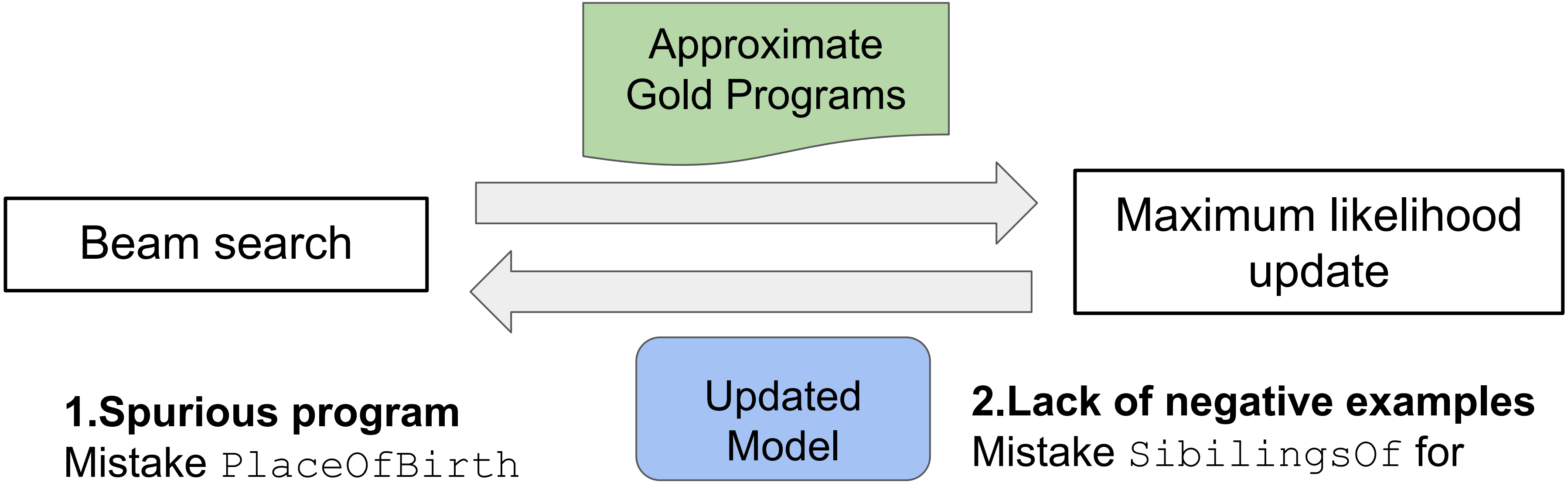


1.**High variance**
Requires a lot of (expensive) samples

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

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

Iterative Maximum Likelihood Training (Hard EM)

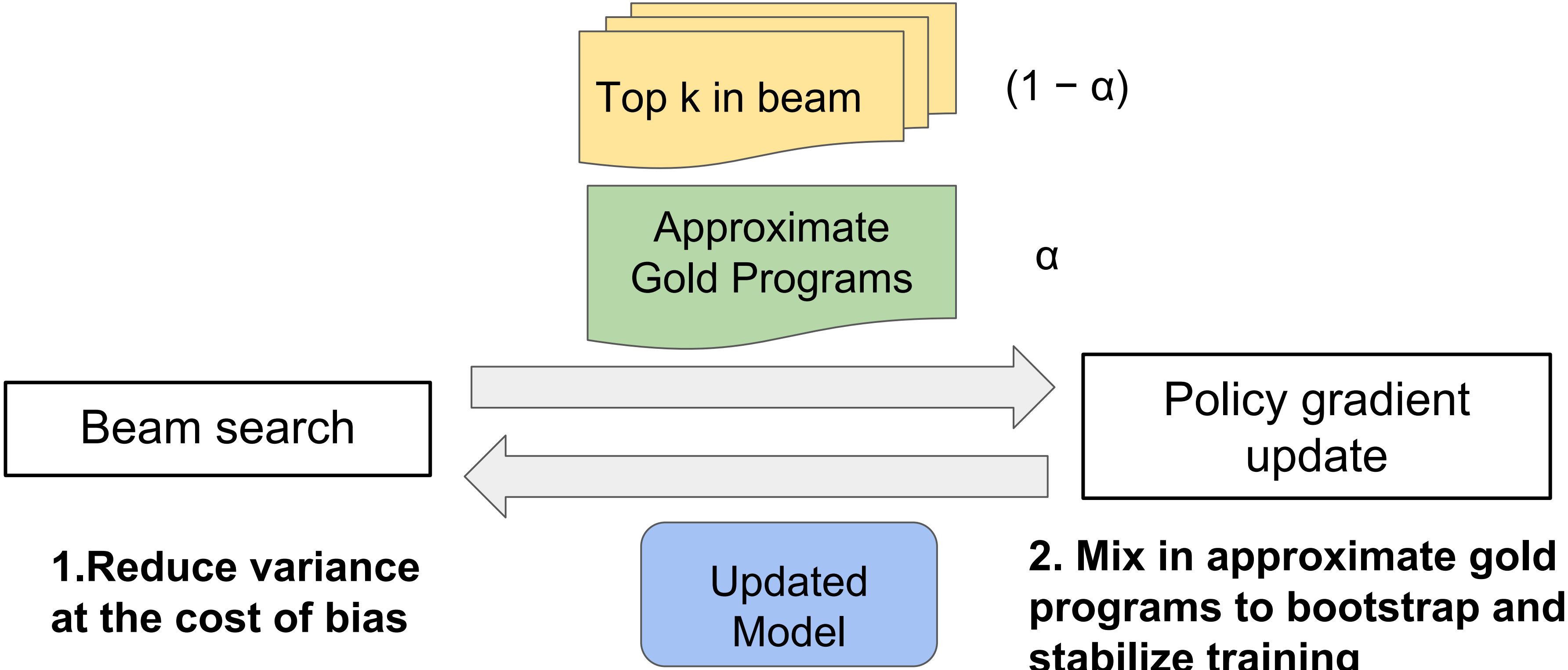


1.Spurious program
Mistake `PlaceOfBirth` for `PlaceOfDeath`.

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

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

Augmented REINFORCE

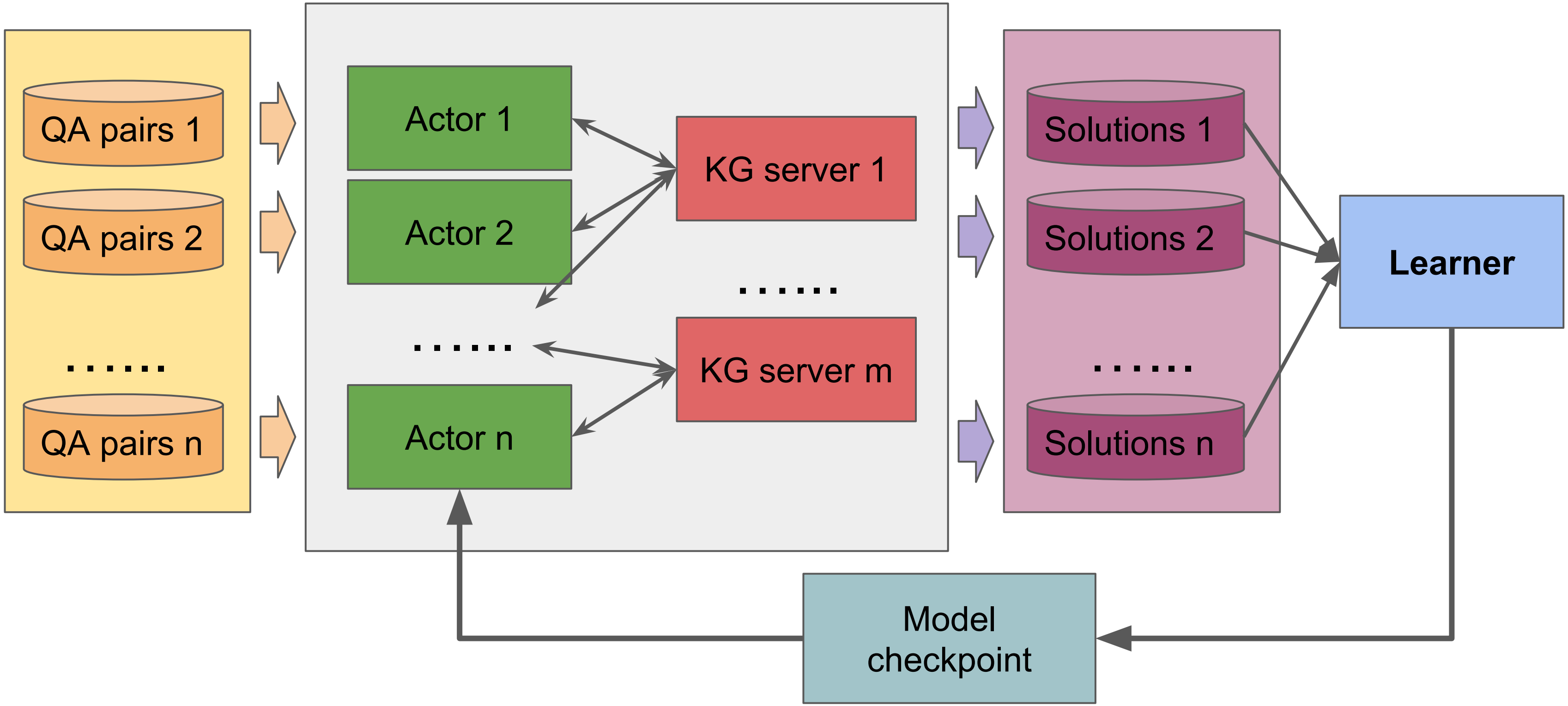


1.Reduce variance at the cost of bias

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

Distributed Architecture

- 200 actors, 100 KG servers, 1 learner



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**

Plan

- Language & control
  - *Neural Symbolic Machines*: Semantic Parsing on Freebase with Weak Supervision
    - Chen Liang, Jonathan Berant, Quoc Le, Kenneth Forbus, Ni Lao

- **Knowledge & scalability**
  - Learning to Organize Knowledge with An *N-Gram Machine*
    - Fan Yang, Jiazhong Nie, William Cohen, Ni Lao

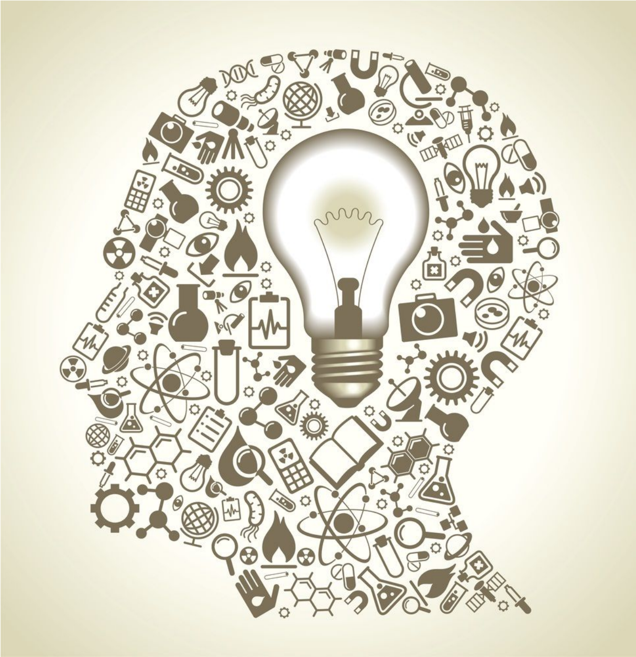
[Wickman 2012]
[Bartol+ 2015]

Where does knowledge come from?

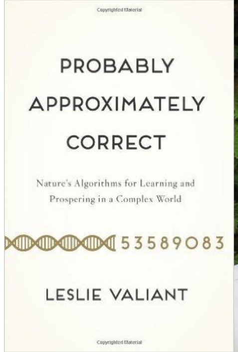
- The human brain contains roughly 100 billion neurons each capable of making around 1,000 connections

- Where do we get these 100 TB parameters?

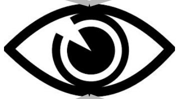
- How many lines of code do I need to write if I want to achieve AI?



The Mind’s Eye



the world



a small machine which can copy large amount of complexity from the world to the brain

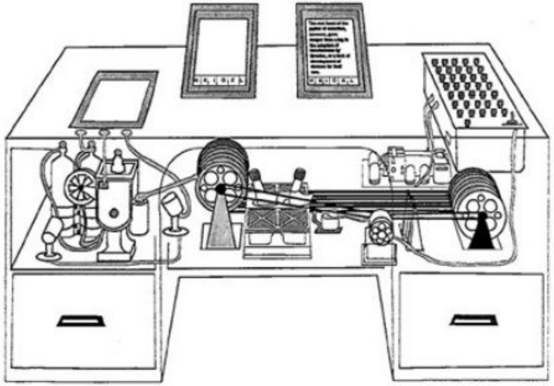
a suitable representation

[Vannevar Bush 1945]

Knowledge and Scalability

- How information should be organized for scalability?

"AS WE MAY THINK"
(1945)



"

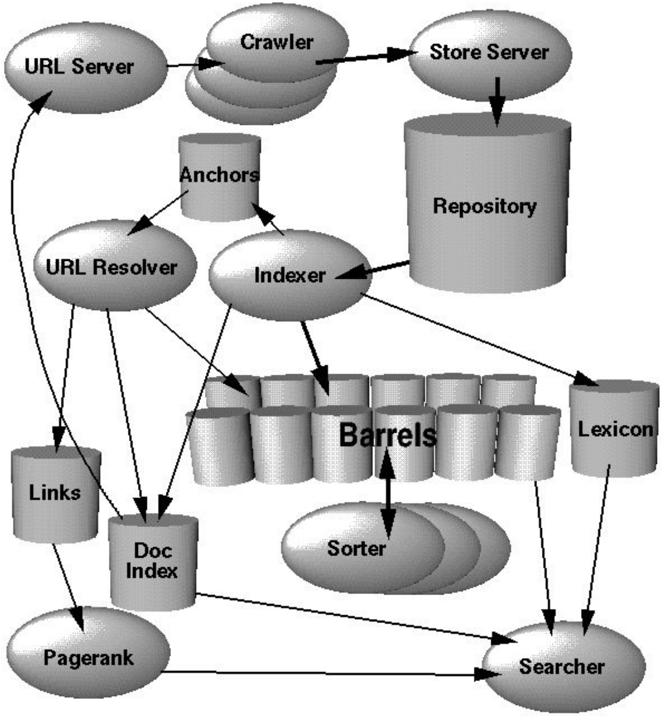
Consider a future device for individual use, which is a sort of mechanized private file and library. It needs a name, and to coin one at random, memex will do. A memex is a device in which an individual stores all his books, records, and communications, and which is mechanized so that it may be consulted with exceeding speed and flexibility. It is an enlarged intimate supplement to his memory.

26

[Brin & Page 1998]

Scalability of modern search engines

- Can respond to user's requests within a fraction of a second
- But are weak at text understanding and complex reasoning



Scalability of mammal memory

- Very rapid adaptation (in just one or a few trials) is necessary for survival
  - E.g., associating taste of food and sickness
- Need fast responses based on large amount of knowledge
  - Needs good representation of knowledge
- However, good representation can only be learnt gradually
  - During sleeps
  - to prevent interference with established associations

[Garcia+ 1966]
[Wickman 2012]
[Bartol+ 2015]

Complementary Learning Theory

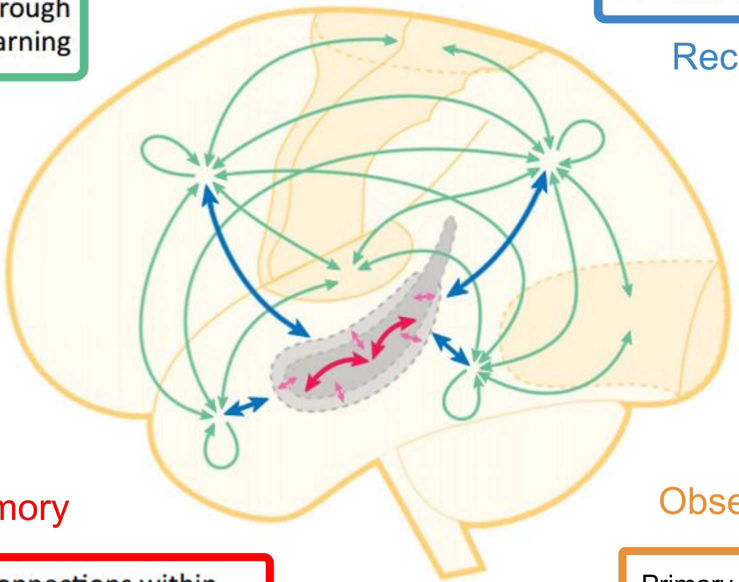
[McClelland+ 1995]
[Kumaran+ 2016]

Connections within and among neocortical areas (green) support gradual acquisition of structured knowledge through interleaved learning

Encoder

Bidirectional connections (blue) link neocortical representations to the hippocampus/MTL for storage, retrieval, and replay

Record & Replay



Episodic memory

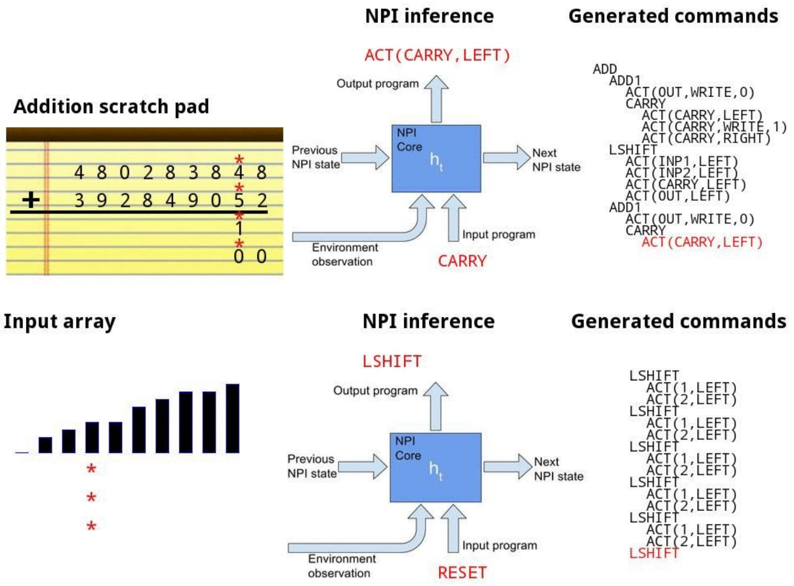
Rapid learning in connections within hippocampus (red) supports initial learning of arbitrary new information

Observations

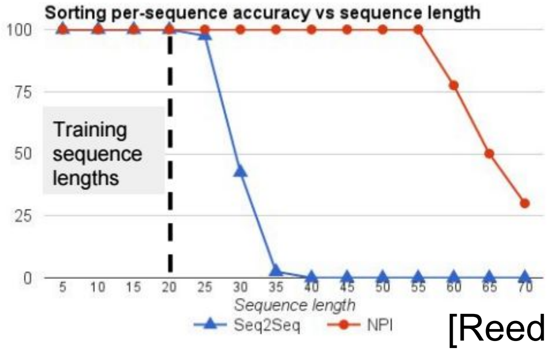
Primary sensory and motor cortices

(Scalable) Neural Nets

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

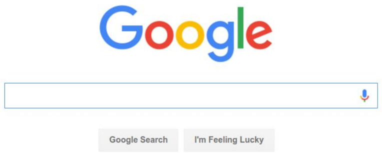


- The learned operations are not as scalable and precise.



[Reed & Freitas 2015]

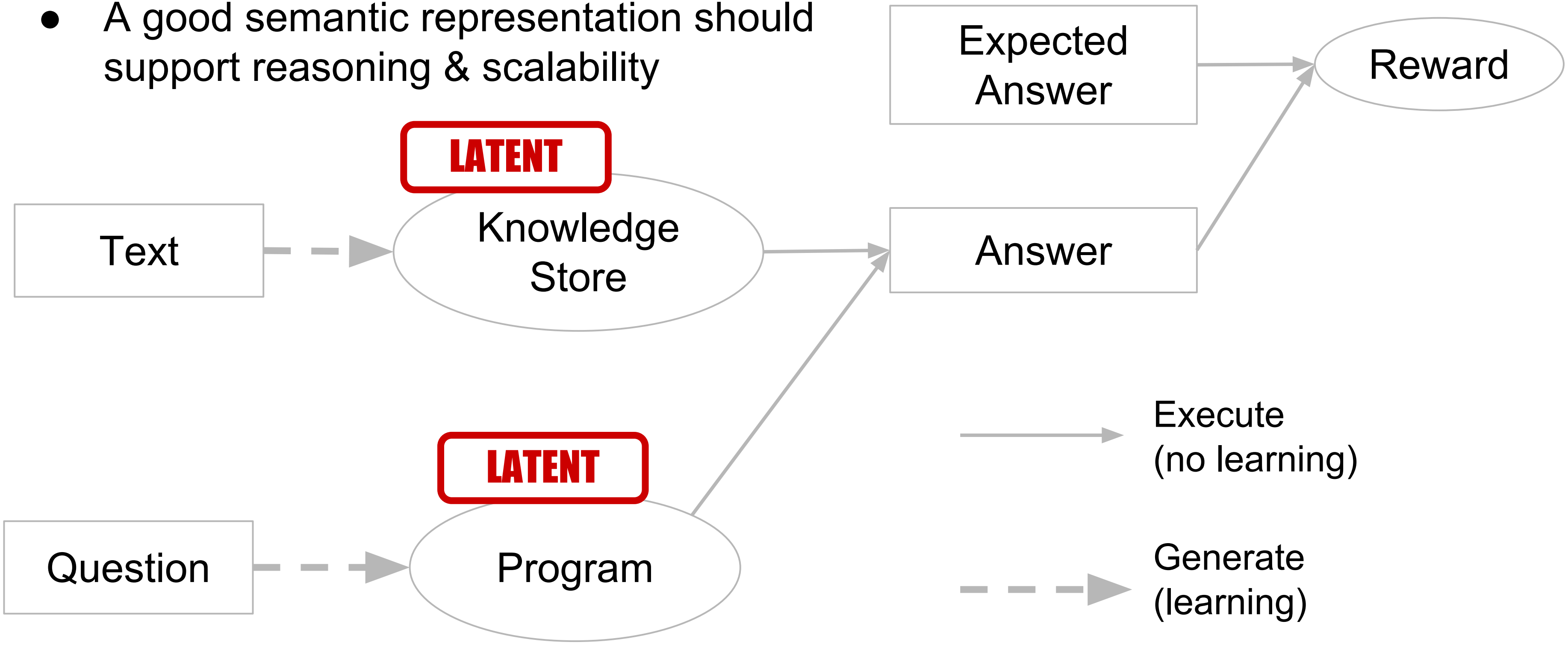
- Why not use existing modules that are scalable, precise and interpretable?



[Zaremba & Sutskever 2016]

Question answering as a simple test bed

- A good semantic representation should support reasoning & scalability



[Weston+ 2015]

Facebook bAbI Tasks

- Simulated question answering tasks to test the ability to "**understand**"
- We introduce a special version ("life-long bAbI"), which has stories of up to 10 million sentences

Sam walks into the kitchen.	Brian is a lion.	Mary journeyed to the den.
Sam picks up an apple.	Julius is a lion.	Mary went back to the kitchen.
Sam walks into the bedroom.	Julius is white.	John journeyed to the bedroom.
Sam drops the apple.	Bernhard is green.	Mary discarded the milk.
Q: Where is the apple?	Q: What color is Brian?	Q: Where was the milk before the den?
A. Bedroom	A. White	A. Hallway

The Knowledge Storage

- Each tuple in the KG is an n-gram, which is a list of symbols
- Each tuple is also associated with a timestamp to reason over time

Table 1: Example of probabilistic knowledge storage. Each sentence may be converted to a distribution over multiple tuples, but only the one with the highest probability is shown here.

Sentences	Knowledge tuples		
	Time stamp	Symbols	Probability
Mary went to the kitchen.	1	`mary to kitchen`	0.9
Mary picked up the milk.	2	`mary the milk`	0.4
John went to the bedroom.	3	`john to bedroom`	0.7
Mary journeyed to the garden.	4	`mary to garden`	0.8

The Programs

- A **program** *C* is a list of **expressions** $c_1 \ldots c_N$, where each c_i is either a special expression *Return* indicating the end of the program, or is of the form $(F, A_1 \ldots A_L)$

Table 2: Functions in N-Gram Machines. The knowledge storage on which the programs can execute is $\mathbf{\Gamma}$, and a knowledge tuple Γ_i is represented as $(i, (\gamma_1, \ldots, \gamma_N))$. "FR" means *from right*.

Name	Inputs	Return
`Hop`	$v_1 \ldots v_L$	$\{\gamma_{L+1} \mid \text{ if } (\gamma_1 \ldots \gamma_L) == (v_1, \ldots, v_L), \forall \Gamma \in \mathbf{\Gamma}\}$
`HopFR`	$v_1 \ldots v_L$	$\{\gamma_{N-L} \mid \text{ if } (\gamma_{N-L+1} \ldots \gamma_N) == (v_L, \ldots, v_1), \forall \Gamma \in \mathbf{\Gamma}\}$
`Argmax`	$v_1 \ldots v_L$	$\text{argmax}_i \{(\gamma_{L+1}, i) \mid \text{ if } (\gamma_1 \ldots \gamma_L) == (v_1, \ldots, v_L), \forall \Gamma_i \in \mathbf{\Gamma}\}$
`ArgmaxFR`	$v_1 \ldots v_L$	$\text{argmax}_i \{(\gamma_{N-L}, i) \mid \text{ if } (\gamma_{N-L+1} \ldots \gamma_N) == (v_L, \ldots, v_1), \forall \Gamma_i \in \mathbf{\Gamma}\}$

Example KS & Program

Table 6: Task 2 Two Supporting Facts

Story	Knowledge Storage
Sandra journeyed to the hallway. John journeyed to the bathroom. Sandra grabbed the football. Daniel travelled to the bedroom. John got the milk. John dropped the milk.	Sandra journeyed hallway John journeyed bathroom Sandra got football Daniel journeyed bedroom John got milk John got milk
Question	**Program**
Where is the milk?	ArgmaxFR milk got Argmax V1 journeyed

CopyNet [Gu+ 2016]
NSM [Liang+ 2017]

Seq2Seq components

- A **knowledge encoder** that converts sentences to knowledge tuples and defines a distribution $P(\Gamma_i | s_i, s_{i-1}; \theta_{\text{enc}})$ $\boxed{s_{i-1} \text{ for co-references}}$
  - The probability of a KS $\Gamma = \{\Gamma_1 \ldots \Gamma_n\}$ is the product of its tuples' probabilities:

$$P(\mathbf{\Gamma} | \mathbf{s}; \theta_{\text{enc}}) = \Pi_{\Gamma_i \in \mathbf{\Gamma}} P(\Gamma_i | s_i, s_{i-1}; \theta_{\text{enc}})$$

- A **knowledge decoder** that converts tuples back to sentences and defines a distribution $P(s_i | \Gamma_i, s_{i-1}; \theta_{\text{dec}})$ -- it will enable unsupervised training

- A **programmer** that converts questions to programs and defines a distribution $P(C | q, \mathbf{\Gamma}; \theta_{\text{prog}})$ $\boxed{\Gamma \text{ for code assist}}$

Inference

Given an example (s, q, a) from our training set, we would like to maximize the **expected reward**

$$O^{QA}(\theta_{\text{enc}}, \theta_{\text{prog}}) = \sum_{\mathbf{\Gamma}} \sum_{C} P(\mathbf{\Gamma}|\mathbf{s}; \theta_{\text{enc}}) P(C|q, \mathbf{\Gamma}; \theta_{\text{prog}}) R(\mathbf{\Gamma}, C, a),$$

For gradient estimations we apply **beam search** instead of **MCMC**, which has huge variances

It leads to a **hard search problem**, which we solve by having
1) a stabilized **auto-encoding** objective to bias the encoder to more interesting hypotheses;

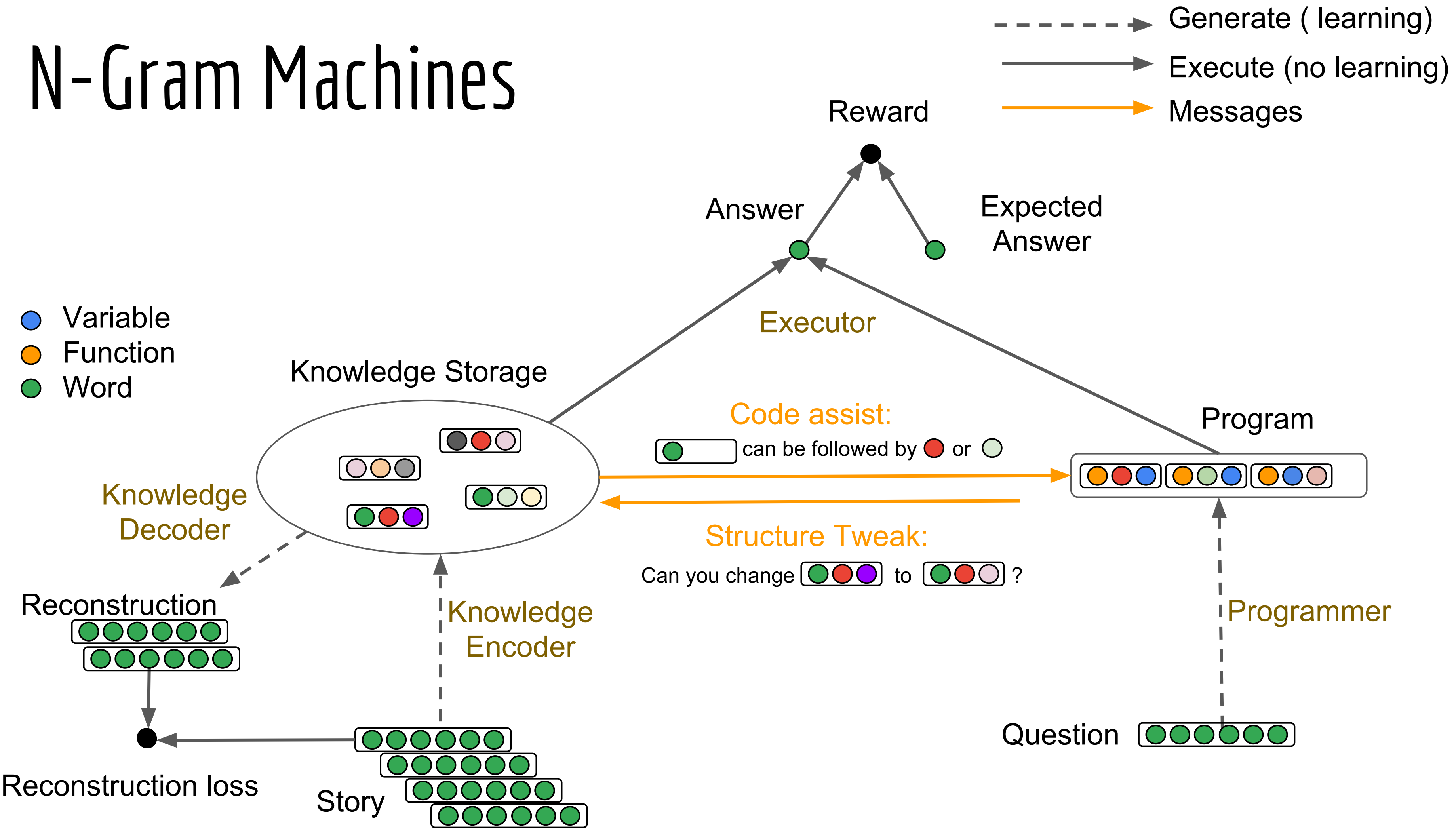
$$O^{\text{AE}}(\theta_{\text{enc}}, \theta_{\text{dec}}) = \mathbb{E}_{p(z|x;\theta_{\text{enc}})}[\log p(x|z; \theta_{\text{dec}})] + \sum_{z \in \mathbf{Z}^N(x)} \log p(x|z; \theta_{\text{dec}}),$$

$Z^N(x)$: all tuples of length N which only consist of words from x

2) a **structural tweak** procedure which retrospectively corrects the inconsistency among multiple hypotheses so that reward can be achieved

- While code assist uses the knowledge storage to inform the programmer,
  structure tweak adjusts the knowledge encoder to cooperate with an uninformed programmer.

N-Gram Machines



Optimization

- For training stability and tweaking, we augment the training objective with **experience replays**

$$\nabla_{\theta_{\text{dec}}} O'(\theta) = \sum_{s_i \in \mathbf{s}} \sum_{\Gamma_i} [\beta(\Gamma_i) + P(\Gamma_i | s_i, s_{i-1}; \theta_{\text{enc}})] \nabla_{\theta_{\text{dec}}} \log P(s_i | \Gamma, s_{i-1}; \theta_{\text{dec}}),$$

$\beta(\Gamma_i)$ is 1 if Γ_i only contains tokens from s_i and 0 otherwise

$$\nabla_{\theta_{\text{enc}}} O'(\theta) = \sum_{s_i \in \mathbf{s}} \sum_{\Gamma_i} [P(\Gamma_i | s_i, s_{i-1}; \theta_{\text{enc}}) \log P(s_i | \Gamma_i, s_{i-1}; \theta_{\text{dec}}) + \mathcal{R}(\mathcal{G}'(\Gamma_i)) + \mathcal{R}(\mathcal{G}(\Gamma_i))] \nabla_{\theta_{\text{enc}}} \log P(\Gamma_i | s_i, s_{i-1}; \theta_{\text{enc}}),$$

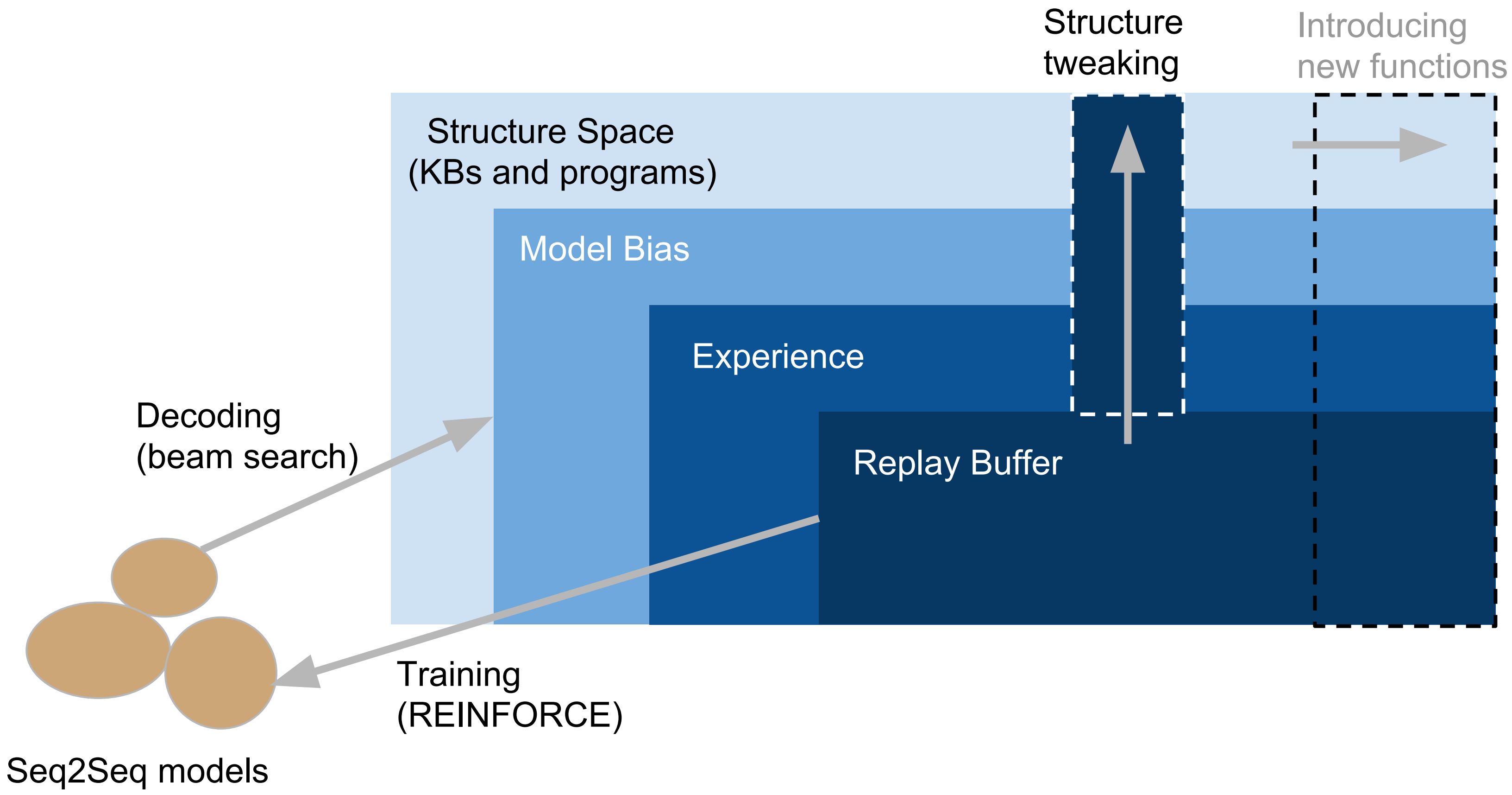
where $\mathcal{R}(\mathcal{G}) = \sum_{\mathbf{\Gamma} \in \mathcal{G}} \sum_C P(\mathbf{\Gamma} | \mathbf{s}; \theta_{\text{enc}}) P(C | q, \mathbf{\Gamma}; \theta_{\text{prog}}) R(\mathbf{\Gamma}, C, a)$ is the total expected reward for a set of valid knowledge stores $\mathcal{G}$, $\mathcal{G}(\Gamma_i)$ is the set of knowledge stores which contains the tuple Γ_i, and $\mathcal{G}'(\Gamma_i)$ is the set of knowledge stores which contains the tuple Γ_i through tweaking.

$$\nabla_{\theta_{\text{prog}}} O'(\theta) = \sum_{\mathbf{\Gamma}} \sum_{C} [\alpha I [C \in \mathcal{C}^*(\mathbf{s}, q)] + P(C | q, \mathbf{\Gamma}; \theta_{\text{prog}})] \cdot P(\mathbf{\Gamma} | \mathbf{s}; \theta_{\text{enc}}) R(\mathbf{\Gamma}, C, a) \nabla_{\theta_{\text{prog}}} \log P(C | q, \mathbf{\Gamma}; \theta_{\text{prog}}),$$

where $\mathcal{C}^*(\mathbf{s}, q)$ is the experience replay buffer for $(\mathbf{s}, q)$. $\alpha = 0.1$ is a constant. During training, the program with the highest weighted reward (i.e. $P(\mathbf{\Gamma} | \mathbf{s}; \theta_{\text{enc}}) R(\mathbf{\Gamma}, C, a)$) is added to the replay buffer.

- optimize by **coordinate ascent** – updating three components in alternation with **REINFORCE**

Learning to search



Results

- The bAbI dataset contains twenty tasks in total. We consider the subset of them that are extractive question answering tasks

Table 3: Test accuracy on bAbI tasks with auto-encoding (AE) and structure tweak (ST)

	Task 1	Task 2	Task 11	Task 15	Task 16
MemN2N	1.000	0.830	0.840	1.000	0.440
QA	0.007	0.027	0.000	0.000	0.098
QA + AE	0.709	0.551	**1.000**	0.246	**1.000**
QA + AE + ST	**1.000**	**0.853**	**1.000**	**1.000**	**1.000**

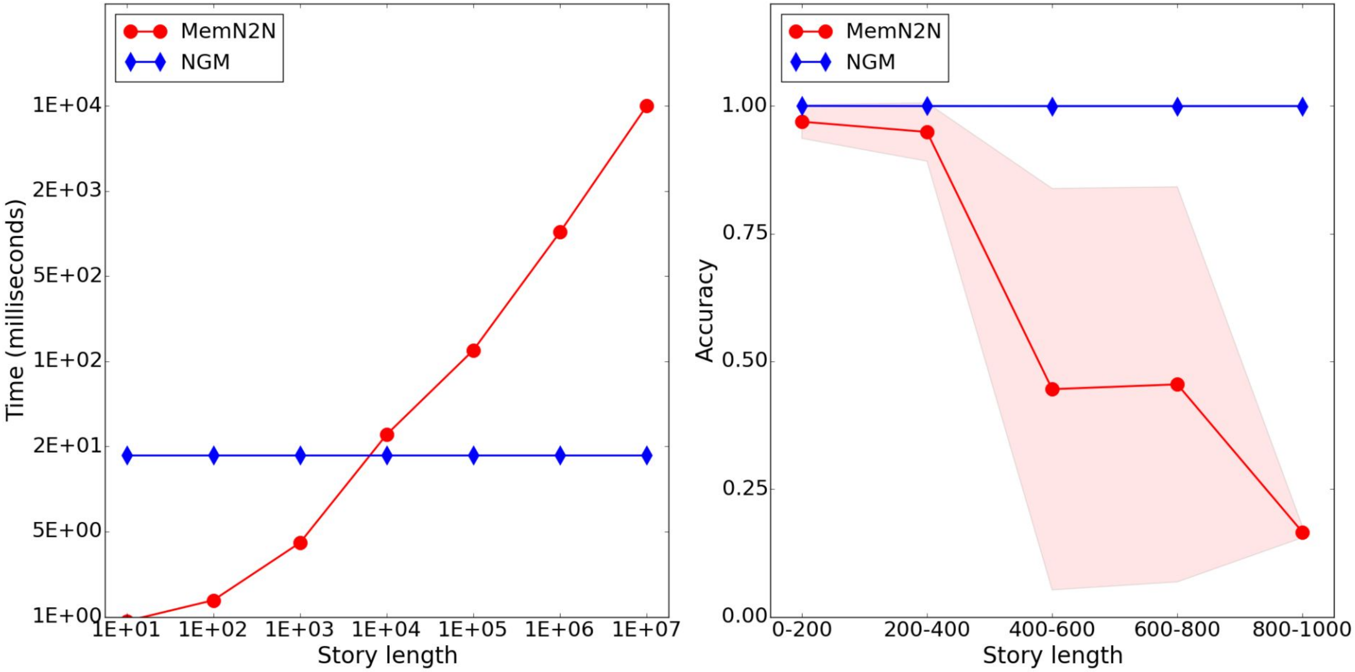
Results

- AE is a strong bias towards good representations
- ST helps to achieve consistency, e.g.,
  - "he" vs "john" (coreference)
  - "cat" vs "cats" (singular vs. plural)
  - "go to" vs "journey to" (synonyms)

QA	QA + AE	QA + AE + ST
went went went	daniel went office	daniel went office
mary mary mary	mary back garden	mary went garden
john john john	john back kitchen	john went kitchen
mary mary mary	mary grabbed football	mary got football
there there there	sandra got apple	sandra got apple
cats cats cats	cats afraid wolves	cat afraid wolves
mice mice mice	mice afraid wolves	mouse afraid wolves
is is cat	gertrude is cat	gertrude is cat

Result

- Scalability



Thanks!

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>
```

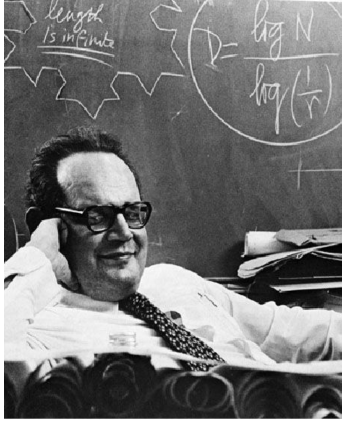
In which

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

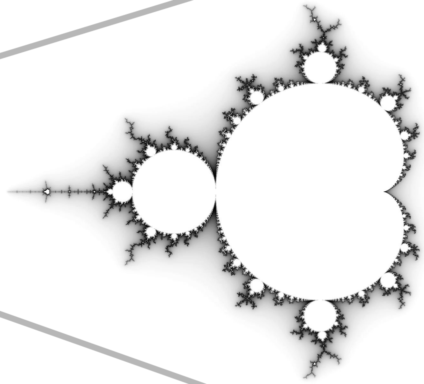
- **Distribution of the length of generated programs**

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

Mandelbrot Set



the nature
of complex
numbers



$$z_0 = 0$$

$$z_{n+1} = z_n^2 + c$$

$$c \in M \iff \lim_{n \to \infty} |z_{n+1}| \leq 2$$

Structure Tweak

- No reward
    - Caused by semantic (i.e. run-time) errors
        - Program: Hop var *cats*
        - Knowledge tuple: *Cat* afraid wolves
    - Propose knowledge tuples that allow the program with high probability to obtain positive reward
- Low expected reward
    - program with high reward in P(prog|q, kg) ≠ program with high probability in P(prog|q)
        - “Hop var *journeyed*” vs “Hop var *went*”
    - Propose knowledge tuples that allow program with high probability to obtain high rewards

Example KS & Program

Table 8: Task 15 Basic Deduction

Story	**Knowledge Storage**
Sheep are afraid of cats.	Sheep afraid cats
Cats are afraid of wolves.	Cat afraid wolves
Jessica is a sheep.	Jessica is sheep
Mice are afraid of sheep.	Mouse afraid sheep
Wolves are afraid of mice.	Wolf afraid mice
Emily is a sheep.	Emily is sheep
Winona is a wolf.	Winona is wolf
Gertrude is a mouse.	Gertrude is mouse
Question	**Program**
What is Emily afraid of?	Hop Emily is Hop V1 afraid

Example KS & Program

Table 9: Task 16 Basic Induction

Story	Knowledge Storage
Berhard is a rhino. Lily is a swan. Julius is a swan. Lily is white. Greg is a rhino. Julius is white. Brian is a lion. Bernhard is gray. Brian is yellow.	Bernhard a rhino Lily a swan Julius a swan Lily is white Greg a rhino Julius is white Brian a lion Bernhard is gray Brian is yellow
Question	**Program**
What color is Greg?	Hop Greg a HopFR V1 a Hop V2 is