Neural Symbolic Machines: Learning Semantic Parsers on Freebase with Weak Supervision

Chen Liang[1], Jonathan Berant[2], Quoc Le, Kenneth Forbus, Ni Lao

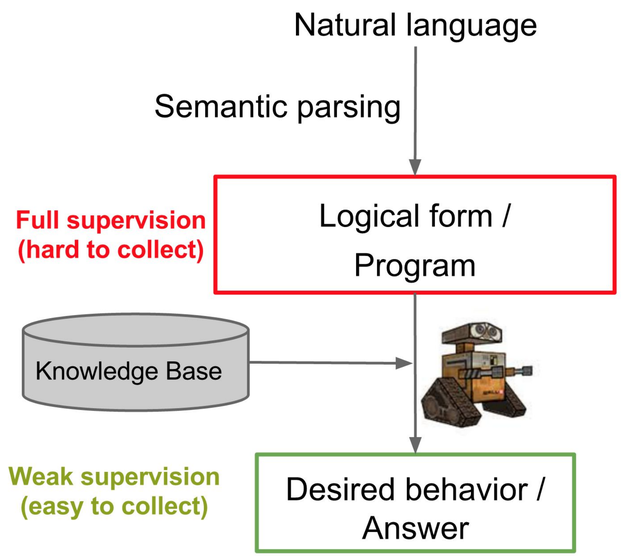
[1]Work done while the author was interning at Google
[2]Work done while the author was a visiting scholar at Google

Research at Google

TEL AVIV UNIVERSITY

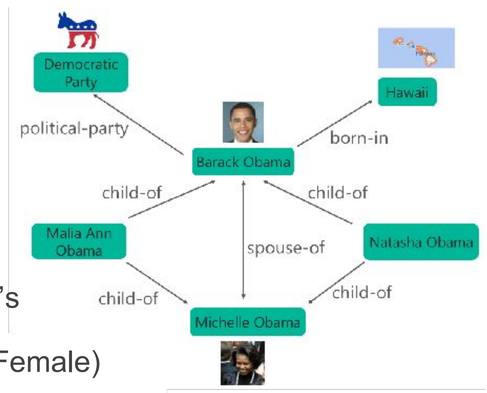
NORTHWESTERN UNIVERSITY

Semantic Parsing



Question Answering with Freebase (WebQuestionsSP Dataset)

- **Large-scale Knowledge Base**
  - Properties of Hundreds of millions of entities Plus relations among them
  - E.g. Freebase, 26k predicates, 200M entities, 3B triples

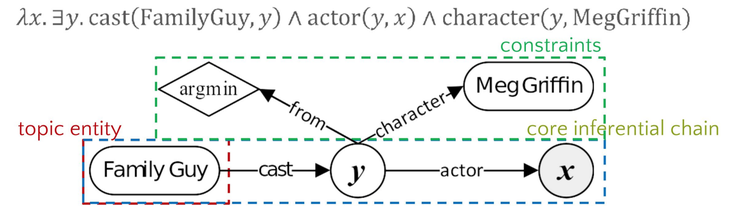


- **Question Answering**
  - "What are the names of Obama's daughters?"
  - λx.parent(Obama,x)∩gender(x,Female)

- **Previous state-of-the-art**
  - Staged Query Graph Generation (STAGG)

Who first voiced Meg on Family Guy?

$\lambda x.\exists y.\text{cast}(\text{FamilyGuy}, y) \wedge \text{actor}(y, x) \wedge \text{character}(y, \text{MegGriffin})$

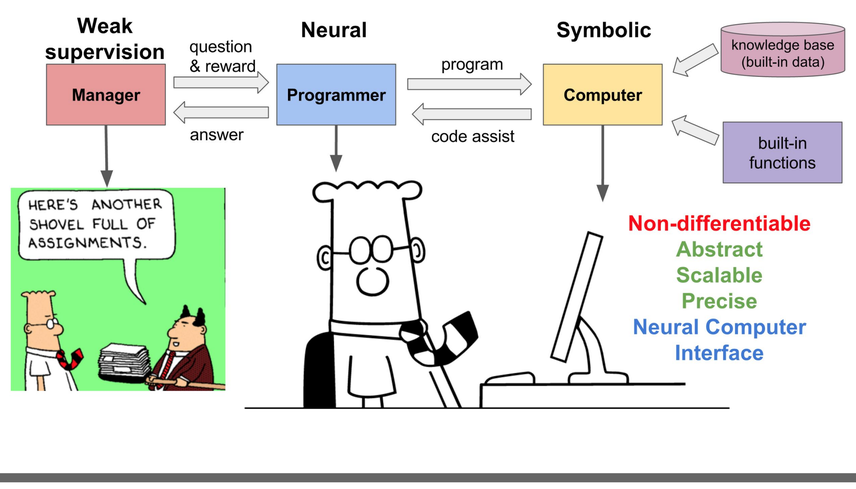


[Yih+ 2016]

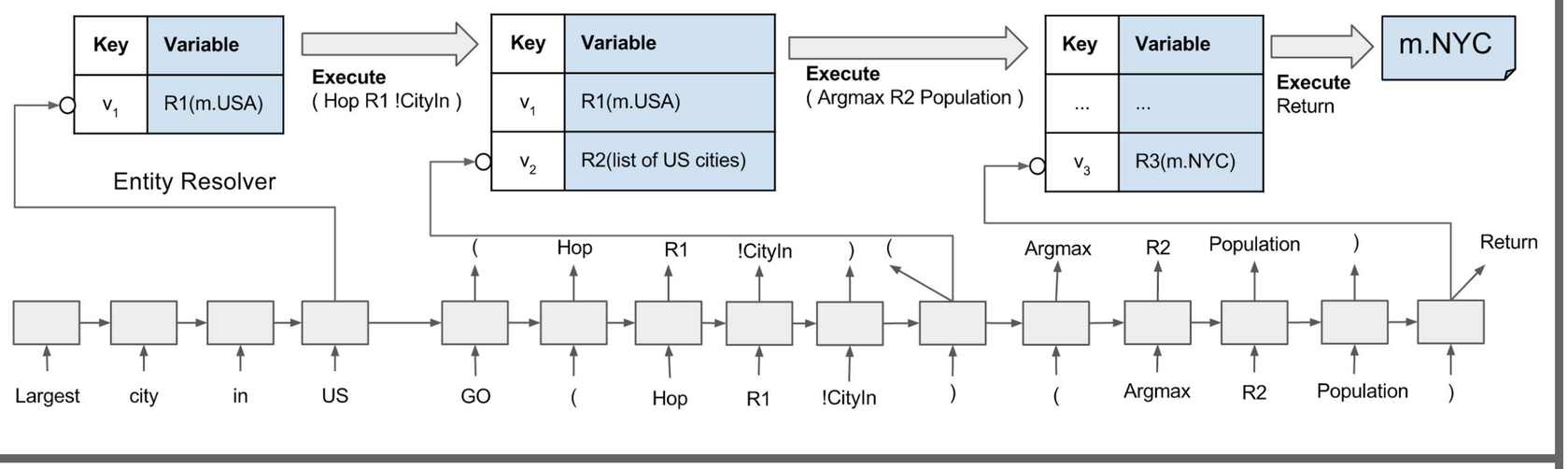
- **Our semantic representation**: Lisp program
  - High-level programming language with uniform syntax
  - Built-in functions to query the KB and process the result

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

MPC Framework



Neural Symbolic Machines with Key-Variable Memory

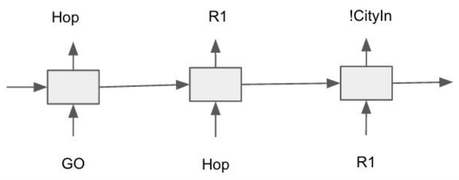


Neural Computer Interface

- A strong IDE / interpreter reduces the search space: exclude the invalid choices that will cause error

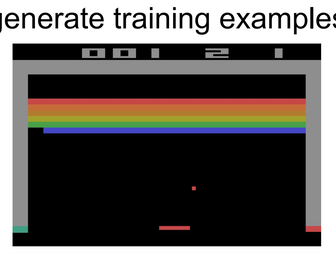
Syntax check: Only a variable can follow 'Hop'

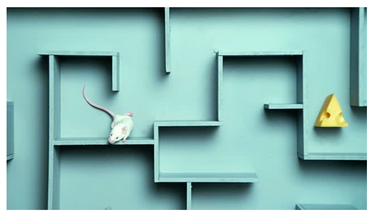
Semantic check: only relations that are connected to R1 can be used **~ 20k => ~ 100**



Sampling v.s. Beam Search

- Programming is deterministic: closer to a maze than Atari game
- Uses beam search (final beam with normalized probabilities) to generate training examples





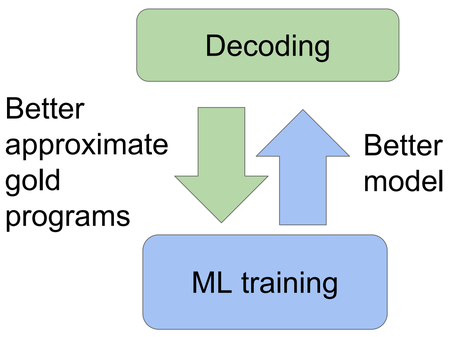
Iterative ML Training

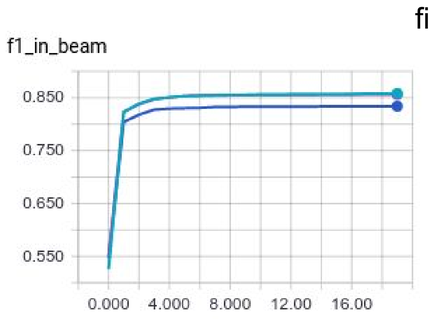
- Optimize log likelihood of **approximate gold programs**

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

- Fast, but suboptimal
  - **Spurious programs**: wrong programs that happen to produce the correct answers, e.g., answering PlaceOfBirth with PlaceOfDeath
  - **Lacking negative examples**: hard to differentiate related relations, e.g., ParentOf, ChildrenOf, SiblingOf.

- Bootstrapping effect





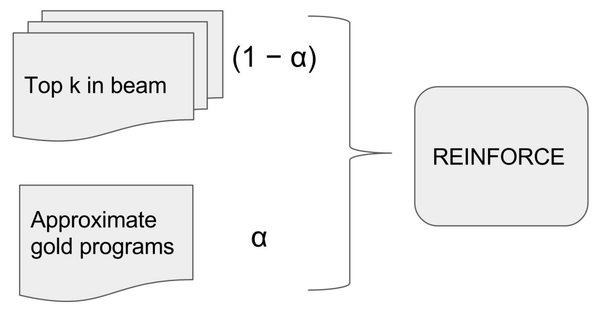
Augmented REINFORCE

- Optimize the **expected F1** of generated programs

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

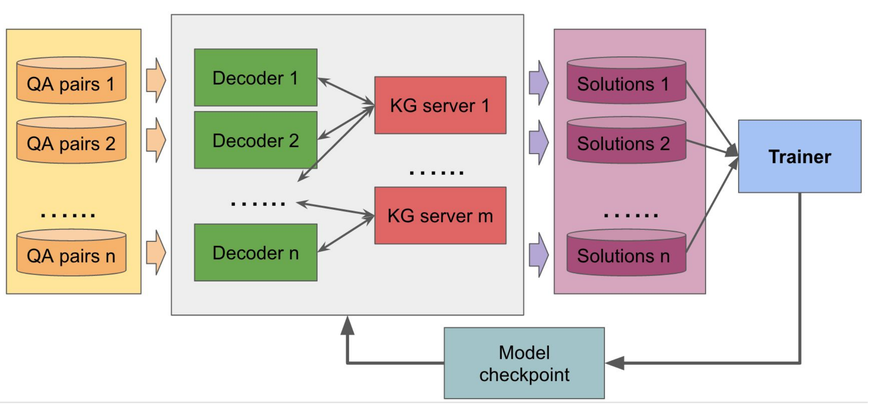
$$\nabla_\theta J^{RL}(\theta) = \sum_q \sum_{a_{0:T}} P(a_{0:T}|q, \theta)[R(q, a_{0:T}) - B(q)] \nabla_\theta \log P(a_{0:T}|q, \theta)$$
$$B(q) = \sum_{a_{0:T}} P(a_{0:T}|q, \theta) R(q, a_{0:T})$$

- **Problem**: slow and stuck at local optima
- **Augmentation**: add approximate gold program into final beam with a reasonably large probability



Implementation

- 200 decoders, 50 KB servers, 1 trainer, 251 machines in total
- Since the network is small, we didn't see much speedup from GPU



Experiments & Analysis

- New state-of-the-art without manual engineering

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

- Curriculum learning in iterative ML training
  - First use only "Hop" and limit program length to 2
  - Then add other functions and limit program length to 3

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

- Comparison of iterative ML, REINFORCE and augmented REINFORCE

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**

- Techniques to 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