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

[Image of diagram showing the process:

Natural Language Question/Instruction → Program / Logical Form → Answer

Full supervision (hard to collect)

Weak supervision (easy to collect)

[Berant, et al 2013; Liang 2013]
Largest city in US?

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

NYC

1. Compositionality

2. Large Search Space

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

- 5,810 questions Google Suggest API & Amazon MTurk
- Remove invalid QA pairs
- 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.

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

[Reed & Freitas 2015]

[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

Weak supervision
Manager

Question
Answer

Neural
Programmer

Program
Output

Symbolic
Computer

Knowledge Base

Predefined Functions

Abstract
Scalable
Precise
Non-differentiable

HERE’S ANOTHER SHOVEL FULL OF ASSIGNMENTS.
Simple Seq2Seq model is not enough

1. Compositionality

2. Large Search Space

23K predicates,
82M entities,
417M triplets
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

<table>
<thead>
<tr>
<th>Key (Embedding)</th>
<th>Variable (Symbol)</th>
<th>Value (Data in Computer)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V_0$</td>
<td>R0</td>
<td>m.USA</td>
</tr>
<tr>
<td>$V_1$</td>
<td>R1</td>
<td>[m.SF, m.NYC, ...]</td>
</tr>
</tbody>
</table>

Expression is finished.

Result

Execution

Computer
Key-Variable Memory: Reuse Intermediate Value

<table>
<thead>
<tr>
<th>Key (Embedding)</th>
<th>Variable (Symbol)</th>
<th>Value (Data in Computer)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V_0$</td>
<td>$R0$</td>
<td>$m.USA$</td>
</tr>
<tr>
<td>$V_1$</td>
<td>$R1$</td>
<td>$[m.SF, m.NYC, ...]$</td>
</tr>
</tbody>
</table>
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

### Decoder Vocab

<table>
<thead>
<tr>
<th>Variable</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V_0$</td>
<td>R0</td>
</tr>
<tr>
<td>$V_1$</td>
<td>R1</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>$E_0$</td>
<td>Hop</td>
</tr>
<tr>
<td>$E_1$</td>
<td>Argmax</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>$P_0$</td>
<td>CityIn</td>
</tr>
<tr>
<td>$P_1$</td>
<td>BornIn</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

- **Variables:** <10
- **Functions:** <10
- **Predicates:** 23K
Last token is ‘(’, so has to output a function name next.
Code Assistance: Semantic Constraint

Decoder Vocab

<table>
<thead>
<tr>
<th>V_0</th>
<th>R0</th>
</tr>
</thead>
<tbody>
<tr>
<td>V_1</td>
<td>R1</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>E_0</td>
<td>Hop</td>
</tr>
<tr>
<td>E_1</td>
<td>Argmax</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>P_0</td>
<td>CityIn</td>
</tr>
<tr>
<td>P_1</td>
<td>BornIn</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Functions: <10

Variables: <10

Predicates: 23K
Given definition of \( \text{Hop} \), need to output a predicate that is connected to \( \text{R}2 \) (\( \text{m.USA} \)).

**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: SiblingsOf for ParentsOf.

\[ J^{ML}(\theta) = \sum_{q} \log P(a_{0:T}^{\text{best}}(q)|q, \theta) \]
Augmented REINFORCE

1. Reduce variance at the cost of bias

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

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

In which

- v0 = “College/University” (m.01y2hn1)
- v1 = “Russell Wilson” (m.05c10yf)

• **Distribution of the length of generated programs**

<table>
<thead>
<tr>
<th>#Expressions</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Percentage</strong></td>
<td>0.4%</td>
<td>62.9%</td>
<td>29.8%</td>
<td>6.9%</td>
</tr>
<tr>
<td><strong>F1</strong></td>
<td>0.0</td>
<td>73.5</td>
<td>59.9</td>
<td>70.3</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Model</th>
<th>Avg. Prec. @1</th>
<th>Avg. Rec. @1</th>
<th>Avg. F1 @1</th>
<th>Acc. @1</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>STAGG</strong></td>
<td>67.3</td>
<td>73.1</td>
<td>66.8</td>
<td>58.8</td>
</tr>
<tr>
<td><strong>NSM – our model</strong></td>
<td>70.8</td>
<td>76.0</td>
<td><strong>69.0</strong></td>
<td>59.5</td>
</tr>
<tr>
<td><strong>STAGG (full supervision)</strong></td>
<td><strong>70.9</strong></td>
<td><strong>80.3</strong></td>
<td>71.7</td>
<td>63.9</td>
</tr>
</tbody>
</table>
Augmented REINFORCE

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

<table>
<thead>
<tr>
<th>Settings</th>
<th>Train Avg. F1@1</th>
<th>Valid Avg. F1@1</th>
</tr>
</thead>
<tbody>
<tr>
<td>iterative ML only</td>
<td>68.6</td>
<td>60.1</td>
</tr>
<tr>
<td>REINFORCE only</td>
<td>55.1</td>
<td>47.8</td>
</tr>
<tr>
<td>Augmented REINFORCE</td>
<td>83.0</td>
<td>67.2</td>
</tr>
</tbody>
</table>
Thanks!

- Manager
  - Question
  - Answer

- Programmer
  - Programs
  - Outputs

- Computer
  - Knowledge Base
  - Predefined Functions

- Neural
- Symbolic

- Key-Variable Memory
- Code Assistance

- Augmented REINFORCE
- Weak supervision
Backup Slides
Semantic Parsing as Program Induction

Learning classifiers

- Hotdog!
- Not hotdog!

Learning programs

Illustration of the DNC architecture

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

- Hotdog!
- Not hotdog!

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 SiblingsOf for ParentsOf.

\[ J^{ML}(\theta) = \sum_q \log P(a_{0:T}^{\text{best}}(q)|q, \theta) \]
Key-Variable Memory: Reuse Intermediate Value

<table>
<thead>
<tr>
<th>Key (Embedding)</th>
<th>Variable (Symbol)</th>
<th>Value (Data in Computer)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V_0$</td>
<td>R0</td>
<td>m.USA</td>
</tr>
<tr>
<td>$V_1$</td>
<td>R1</td>
<td>m.SF, m.NYC, ...</td>
</tr>
</tbody>
</table>

Softmax

Argmax

Argmax

!CityIn
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.01y2hn1)
  v1 = “Russell Wilson” (m.05c10yf)

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

<table>
<thead>
<tr>
<th>#Expressions</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Percentage</strong></td>
<td>0.4%</td>
<td>62.9%</td>
<td>29.8%</td>
<td>6.9%</td>
</tr>
<tr>
<td><strong>F1</strong></td>
<td>0.0</td>
<td>73.5</td>
<td>59.9</td>
<td>70.3</td>
</tr>
</tbody>
</table>
1. **High variance**
   Requires a lot of (expensive) samples

2. **Bootstrap problem**
   Small 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

Repeat

1. Spurious program
   Mistake PlaceOfBirth for PlaceOfDeath.

2. Lack of negative examples
   Mistake SibilingsOf for ParentsOf.

Reward-Augmented Beam Search

Actor

Learner

Maximum Likelihood

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

Approximate Gold Programs
Augmented REINFORCE

Repeat

Beam Search
Reduce variance at the cost of bias

Actor

Learner

Policy gradient
Mix in approximate gold programs to bootstrap and stabilize training

Top k in beam

Approximate gold programs

$(1 - \alpha)$

$\alpha$
Future Work

- Define new functions
- Read the output
More Ablation Analysis

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

<table>
<thead>
<tr>
<th>Settings</th>
<th>Avg. Prec.@Best</th>
<th>Avg. Rec.@Best</th>
<th>Avg. F1@Best</th>
<th>Acc. @Best</th>
</tr>
</thead>
<tbody>
<tr>
<td>No curriculum</td>
<td>79.1</td>
<td>91.1</td>
<td>78.5</td>
<td>67.2</td>
</tr>
<tr>
<td>Curriculum</td>
<td>88.6</td>
<td>96.1</td>
<td>89.5</td>
<td>79.8</td>
</tr>
</tbody>
</table>

- **Reduce overfitting**

<table>
<thead>
<tr>
<th>Settings</th>
<th>Δ Avg. F1@1</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Pretrained word embeddings</td>
<td>-5.5</td>
</tr>
<tr>
<td>- Pretrained relation embeddings</td>
<td>-2.7</td>
</tr>
<tr>
<td>- Dropout on GRU input and output</td>
<td>-2.4</td>
</tr>
<tr>
<td>- Dropout on softmax</td>
<td>-1.1</td>
</tr>
<tr>
<td>- Anonymize entity tokens</td>
<td>-2.0</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Settings</th>
<th>Avg. Prec.@Best</th>
<th>Avg. Rec.@Best</th>
<th>Avg. F1@Best</th>
<th>Acc.@Best</th>
</tr>
</thead>
<tbody>
<tr>
<td>No curriculum</td>
<td>79.1</td>
<td>91.1</td>
<td>78.5</td>
<td>67.2</td>
</tr>
<tr>
<td>Curriculum</td>
<td>88.6</td>
<td>96.1</td>
<td>89.5</td>
<td>79.8</td>
</tr>
</tbody>
</table>

Inspired by STAGG [Yih, et al 2016]
Augmented REINFORCE

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

Manager → Programmer ↓ question
            Programmer → Computer ← inputs&code
            Computer → Manager ← outputs

Knowledge Base

Predefined Functions

1. Key-Variable Memory
2. Code Assistance
3. Augmented REINFORCE

Future Work

Programmer → Computer ↓ Define new functions
            Computer → Programmer ← Save new knowledge
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.

- 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
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

<table>
<thead>
<tr>
<th>Settings</th>
<th>Avg. Prec.@Best</th>
<th>Avg. Rec.@Best</th>
<th>Avg. F1@Best</th>
<th>Acc.@Best</th>
</tr>
</thead>
<tbody>
<tr>
<td>No curriculum</td>
<td>79.1</td>
<td>91.1</td>
<td>78.5</td>
<td>67.2</td>
</tr>
<tr>
<td>Curriculum</td>
<td>88.6</td>
<td>96.1</td>
<td>89.5</td>
<td>79.8</td>
</tr>
</tbody>
</table>

Inspired by STAGG [Yih, et al 2016]
Reduce Overfitting

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

<table>
<thead>
<tr>
<th>Settings</th>
<th>Δ Avg. F1@1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pretrained word embeddings</td>
<td>-5.5</td>
</tr>
<tr>
<td>Pretrained relation embeddings</td>
<td>-2.7</td>
</tr>
<tr>
<td>Dropout on GRU input and output</td>
<td>-2.4</td>
</tr>
<tr>
<td>Dropout on softmax</td>
<td>-1.1</td>
</tr>
<tr>
<td>Anonymize entity tokens</td>
<td>-2.0</td>
</tr>
</tbody>
</table>
Overview

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

- 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

Mean grid spacing for all modules (M1–M4) in all animals (colour-coded)
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 $(e_1, p, e_2) \in E \times P \times E$
    e.g., (ABELINCOLN, PLACEOFBIRTH, HODGENVILLE)

- **Semantic parsing**
  - Given a knowledge base $K$, and a question $q = (w_1, w_2, \ldots, w_k)$,
  - 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 $(c_1...c_l)$
  - An expression is either a special token "Return" or a list "( F A0 ... Ak )"
  - $F$ is one of the functions

\[
\begin{align*}
(\text{Hop } v \ p) & \Rightarrow \{e_2|e_1 \in v, (e_1, p, e_2) \in \mathbb{K}\} \\
(\text{ArgMax } v \ p) & \Rightarrow \{e_1|e_1 \in v, \exists e_2 \in \mathcal{E}: (e_1, p, e_2) \in \mathbb{K}, \forall e : (e_1, p, e) \in \mathbb{K}, e_2 \geq e\} \\
(\text{ArgMin } v \ p) & \Rightarrow \{e_1|e_1 \in v, \exists e_2 \in \mathcal{E}: (e_1, p, e_2) \in \mathbb{K}, \forall e : (e_1, p, e) \in \mathbb{K}, e_2 \leq e\} \\
(\text{Equal } v_1 \ v_2 \ p) & \Rightarrow \{e_1|e_1 \in v_1, \exists e_2 \in v_2: (e_1, p, e_2) \in \mathbb{K}\}
\end{align*}
\]

- An argument $A_i$ 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
Question: Largest city in US

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

Output: NYC
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]
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

<table>
<thead>
<tr>
<th>Comments</th>
<th>Variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entity extracted from the word after “city in”</td>
<td>R1(m.USA)</td>
</tr>
<tr>
<td>Generated by querying v1 with !CityIn</td>
<td>R2(a list of US cities)</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Embeddings</th>
<th>Variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0.1, -0.2, 0.3, …]</td>
<td>R1(m.USA)</td>
</tr>
<tr>
<td>[0.8, 0.5, -0.3, …]</td>
<td>R2(a list of US cities)</td>
</tr>
</tbody>
</table>

Diagram:
- Input: Largest city
- Intermediate: GO
- Output: !CityIn
- Argmax
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 \( \sim 20k \Rightarrow \sim 100 \)

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: $15k + 30k + 15k \times 2 + 5k = 80k$ params
- Dot product attention
- Pretrained embeddings
- Dropout (a lot)

Encoder
- GRU: $2 \times 50 \times 3 \times 50 = 15k$
- Linear Projection: $300 \times 50 = 15k$
- GloVe

Decoder
- GRU: $2 \times 50 \times 3 \times 50 = 15k$
- Linear Projection: $100 \times 50 = 5k$
- Dropout
- GloVe
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.
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:
  \[ J_{ML}^{\text{ML}}(\theta) = \sum_{q} \log P(a_{0:T}^{\text{best}}(q) | q, \theta) \]

- 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 $\alpha$, and the probabilities of the original programs in the beam are normalized to be $(1 - \alpha)$.
- 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 $\alpha$, 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 $\theta$ randomly $\Rightarrow$ 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 $1$ to $|\mathbb{C}|$ 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 $\theta$ randomly $\Rightarrow$ 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 $1$ to $|\mathbb{C}|$ 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 $1$ to $|\mathbb{C}|$ 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

<table>
<thead>
<tr>
<th>Model</th>
<th>Avg. Prec. @1</th>
<th>Avg. Rec. @1</th>
<th>Avg. F1 @1</th>
<th>Acc. @1</th>
</tr>
</thead>
<tbody>
<tr>
<td>STAGG</td>
<td>67.3</td>
<td>73.1</td>
<td>66.8</td>
<td>58.8</td>
</tr>
<tr>
<td>NSM – our model</td>
<td>70.8</td>
<td>76.0</td>
<td>69.0</td>
<td>59.5</td>
</tr>
<tr>
<td>STAGG (full supervision)</td>
<td>70.9</td>
<td>80.3</td>
<td>71.7</td>
<td>63.9</td>
</tr>
</tbody>
</table>
Augmented REINFORCE

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

<table>
<thead>
<tr>
<th>Settings</th>
<th>Train Avg. F1@1</th>
<th>Valid Avg. F1@1</th>
</tr>
</thead>
<tbody>
<tr>
<td>iterative ML only</td>
<td>68.6</td>
<td>60.1</td>
</tr>
<tr>
<td>REINFORCE only</td>
<td>55.1</td>
<td>47.8</td>
</tr>
<tr>
<td>Augmented REINFORCE</td>
<td>83.0</td>
<td>67.2</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Settings</th>
<th>Avg. Prec.@Best</th>
<th>Avg. Rec.@Best</th>
<th>Avg. F1@Best</th>
<th>Acc.@Best</th>
</tr>
</thead>
<tbody>
<tr>
<td>No curriculum</td>
<td>79.1</td>
<td>91.1</td>
<td>78.5</td>
<td>67.2</td>
</tr>
<tr>
<td>Curriculum</td>
<td>88.6</td>
<td>96.1</td>
<td>89.5</td>
<td>79.8</td>
</tr>
</tbody>
</table>

Inspired by STAGG [Yih, et al 2016]
Reduce Overfitting

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

<table>
<thead>
<tr>
<th>Settings</th>
<th>Δ Avg. F1@1</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Pretrained word embeddings</td>
<td>-5.5</td>
</tr>
<tr>
<td>- Pretrained relation embeddings</td>
<td>-2.7</td>
</tr>
<tr>
<td>- Dropout on GRU input and output</td>
<td>-2.4</td>
</tr>
<tr>
<td>- Dropout on softmax</td>
<td>-1.1</td>
</tr>
<tr>
<td>- Anonymize entity tokens</td>
<td>-2.0</td>
</tr>
</tbody>
</table>
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