Logical reasoning over text

Hanlin

- Reasoning over Structured KB/KG
- Reasoning over Unstructured Text
- Reasoning via Program Induction

Reasoning over structured KB/KG

Case1: Go for a Walk and Arrive at the Answer: Reasoning Over Knowledge Bases with Reinforcement Learning

- Goal: Automatically learn reasoning paths in KBs.

  “Who did Malala Yousafzai share her Nobel Peace prize with?”

  Reasoning path: Malala Yousafzai → WonAward → Nobel Peace Prize 2014 →

  AwardedTo → Kailash Satyarthi.

- Frame the learning problem as one of query answering, that is to say, answering questions of the form (Malala Yousafzai, SharesNobelPrizeWith, ?).

- Efficiently searching the graph for answer-providing paths using reinforcement learning (RL) conditioned on the input question

Go for a Walk and Arrive at the Answer: Reasoning Over Knowledge Bases with Reinforcement Learning. Rajarshi Das*, Andrew McCallum. International Conference on Learning Representations (ICLR) 2018.

RL formulation - POMDP

States. The state space $\mathcal{S}$ consists of all valid combinations in $\mathcal{E} \times \mathcal{E} \times \mathcal{R} \times \mathcal{E}$. Intuitively, we want a state to encode the query $(\mathrm{e}_{1\mathrm{q}}, \mathrm{r}_\mathrm{q})$, the answer $(\mathrm{e}_{2\mathrm{q}})$, and a location of exploration e_t (current location of the RL agent). Thus overall a state $S \in \mathcal{S}$ is represented by $S = (\mathrm{e}_\mathrm{t}, \mathrm{e}_{1\mathrm{q}}, \mathrm{r}_\mathrm{q}, \mathrm{e}_{2\mathrm{q}})$ and the state space consists of all valid combinations.

Unobserved

Observations. The complete state of the environment is not observed. Intuitively, the agent knows its current location (e_t) and $(\mathrm{e}_{1\mathrm{q}}, \mathrm{r}_\mathrm{q})$, but not the answer $(\mathrm{e}_{2\mathrm{q}})$, which remains hidden. Formally, the observation function $\mathcal{O} : \mathcal{S} \to \mathcal{E} \times \mathcal{E} \times \mathcal{R}$ is defined as $\mathcal{O}(s = (\mathrm{e}_\mathrm{t}, \mathrm{e}_{1\mathrm{q}}, \mathrm{r}_\mathrm{q}, \mathrm{e}_{2\mathrm{q}})) = (\mathrm{e}_\mathrm{t}, \mathrm{e}_{1\mathrm{q}}, \mathrm{r}_\mathrm{q})$.

Actions. The set of possible actions $\mathcal{A}_S$ from a state $S = (\mathrm{e}_\mathrm{t}, \mathrm{e}_{1\mathrm{q}}, \mathrm{r}_\mathrm{q}, \mathrm{e}_{2\mathrm{q}})$ consists of all outgoing edges of the vertex e_t in $\mathcal{G}$. Formally $\mathcal{A}_S = \{(\mathrm{e}_\mathrm{t}, r, v) \in E : S = (\mathrm{e}_\mathrm{t}, \mathrm{e}_{1\mathrm{q}}, \mathrm{r}_\mathrm{q}, \mathrm{e}_{2\mathrm{q}}), r \in \mathcal{R}, v \in V\} \cup \{(s, \varnothing, s)\}$.

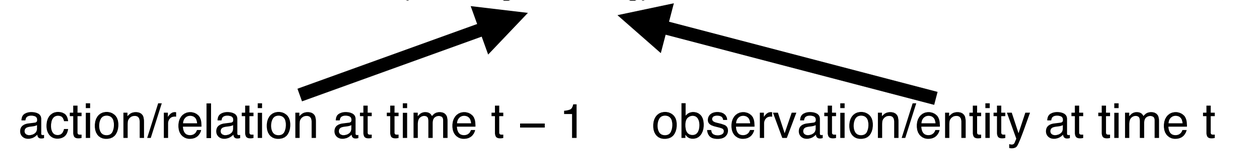
Transition. The environment evolves deterministically by just updating the state to the new vertex incident to the edge selected by the agent. The query and answer remains the same. Formally, the transition function is $\delta : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ defined by $\delta(S, A) = (v, \mathrm{e}_{1\mathrm{q}}, \mathrm{r}_\mathrm{q}, \mathrm{e}_{2\mathrm{q}})$, where $S = (\mathrm{e}_\mathrm{t}, \mathrm{e}_{1\mathrm{q}}, \mathrm{r}_\mathrm{q}, \mathrm{e}_{2\mathrm{q}})$ and $A = (\mathrm{e}_\mathrm{t}, r, v))$.

Rewards. We only have a terminal reward of +1 if the current location is the correct answer at the end and 0 otherwise. To elaborate, if $S_T = (\mathrm{e}_\mathrm{t}, \mathrm{e}_{1\mathrm{q}}, \mathrm{r}_\mathrm{q}, \mathrm{e}_{2\mathrm{q}})$ is the final state, then we receive a reward of +1 if $\mathrm{e}_\mathrm{t} = \mathrm{e}_{2\mathrm{q}}$ else 0., i.e. $R(S_T) = \mathbb{I}\{\mathrm{e}_\mathrm{t} = \mathrm{e}_{2\mathrm{q}}\}$.

Policy Network and Training

An agent based on LSTM encodes the history H_t as a continuous vector $\mathbf{h_t} \in \mathbb{R}^{2d}$. We also have embedding matrix $\mathbf{r} \in \mathbb{R}^{|\mathcal{R}| \times d}$ and $\mathbf{e} \in \mathbb{R}^{|\mathcal{E}| \times d}$ for the binary relations and entities respectively. The history embedding for $H_t = (H_{t-1}, A_{t-1}, O_t)$ is updated according to LSTM dynamics:

$$\mathbf{h_t} = \text{LSTM}(\mathbf{h_{t-1}}, [\mathbf{a_{t-1}}; \mathbf{o_t}]) \tag{1}$$



- The policy network makes the decision to choose an action from all available actions (A_{S_t}) conditioned on the query relation

- Randomized non-stationary history-dependent policy $\pi = (d_1, d_2, \ldots, d_{T-1})$, where $d_t : H_t \rightarrow \mathbf{P}(\mathbf{A}_{S_t})$ and history $H_t = (H_{t-1}, A_{t-1}, O_t)$ is the sequence of observations and actions taken

- For the policy network (π_θ) described above, we want to find parameters θ that maximize the expected reward:

$$J(\theta) = \mathbb{E}_{(e_1,r,e_2)\sim D}\mathbb{E}_{A_1,..,A_{T-1}\sim\pi_\theta}[R(S_T)|S_1 = (e_1, e_1, r, e_2)],$$

Case2: Markov Logic Network and variants

- Goal: Learning to complete structured KB with existing knowledge and logic rules

- Task: Knowledge Base Completion/Predicate Classification
  E.g.: (A, FriendOf, ?); (A, FriendOf, B)?

- In the **E-step** (**Inference step**), we are minimizing the KL divergence between the variational posterior distribution $Q_\theta (H|O)$ and the true posterior distribution $P_w (H|O)$

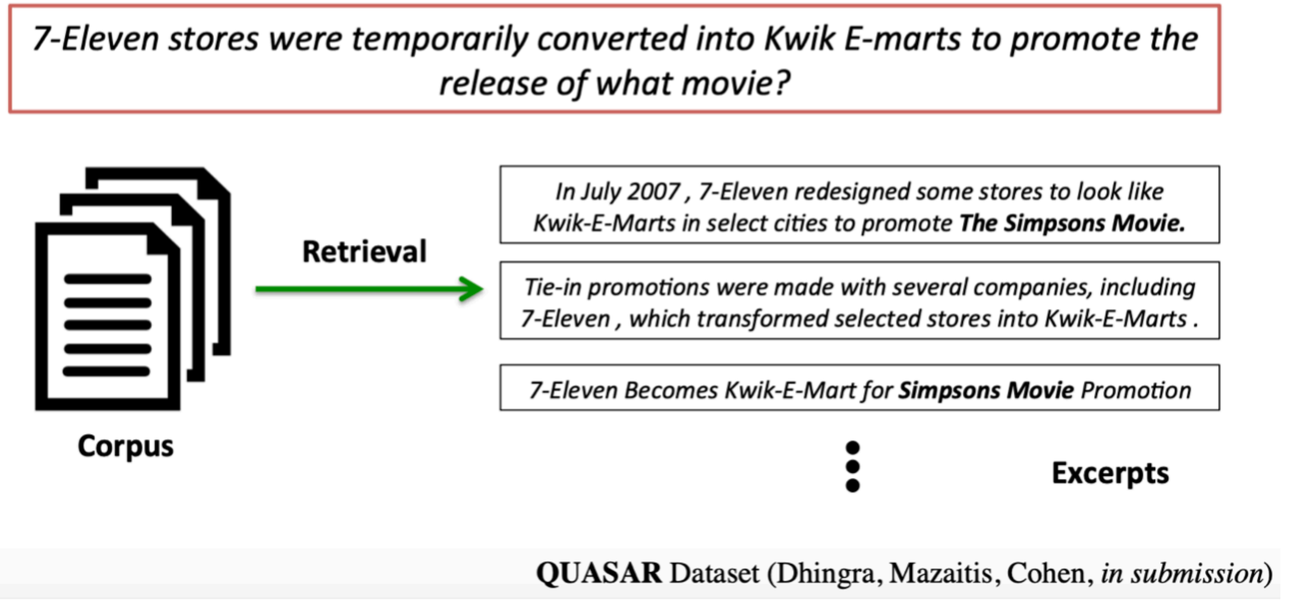
- In the **M-step** (**learning step**), we are learning the weights of logic formulae in Markov Logic Networks with the variational posterior $Q_\theta (H|O)$ fixed

Can Graph Neural Networks Help Logic Reasoning? Yuyu Zhang, Le Song etc 2020 ICLR
pLogicNet Meng Qu, Jian Tang 2020 ICLR

Reasoning over Unstructured Text

Features of Multi-hop Reasoning

- Requires finding and reasoning over **multiple supporting documents** to answer;
- Not constrained to any pre-existing KBs or knowledge schemas



QUASAR Dataset (Dhingra, Mazaitis, Cohen, *in submission*)

Case1: Learning to Retrieve Reasoning Paths over Wikipedia Graph for Question Answering

- Retrieves reasoning paths over the Wikipedia graph to answer multi-hop open-domain questions

- Given a question q, the framework aims at deriving its answer a by **retrieving and reading reasoning paths**, each of which is represented with a sequence of paragraphs: E = [pi , . . . , pk].

- Formulate the task by decomposing the objective into the **retriever objective** S_retr(q, E) that selects reasoning paths E relevant to the question, and the **reader objective** S_read(q, E, a) that finds the answer a in E

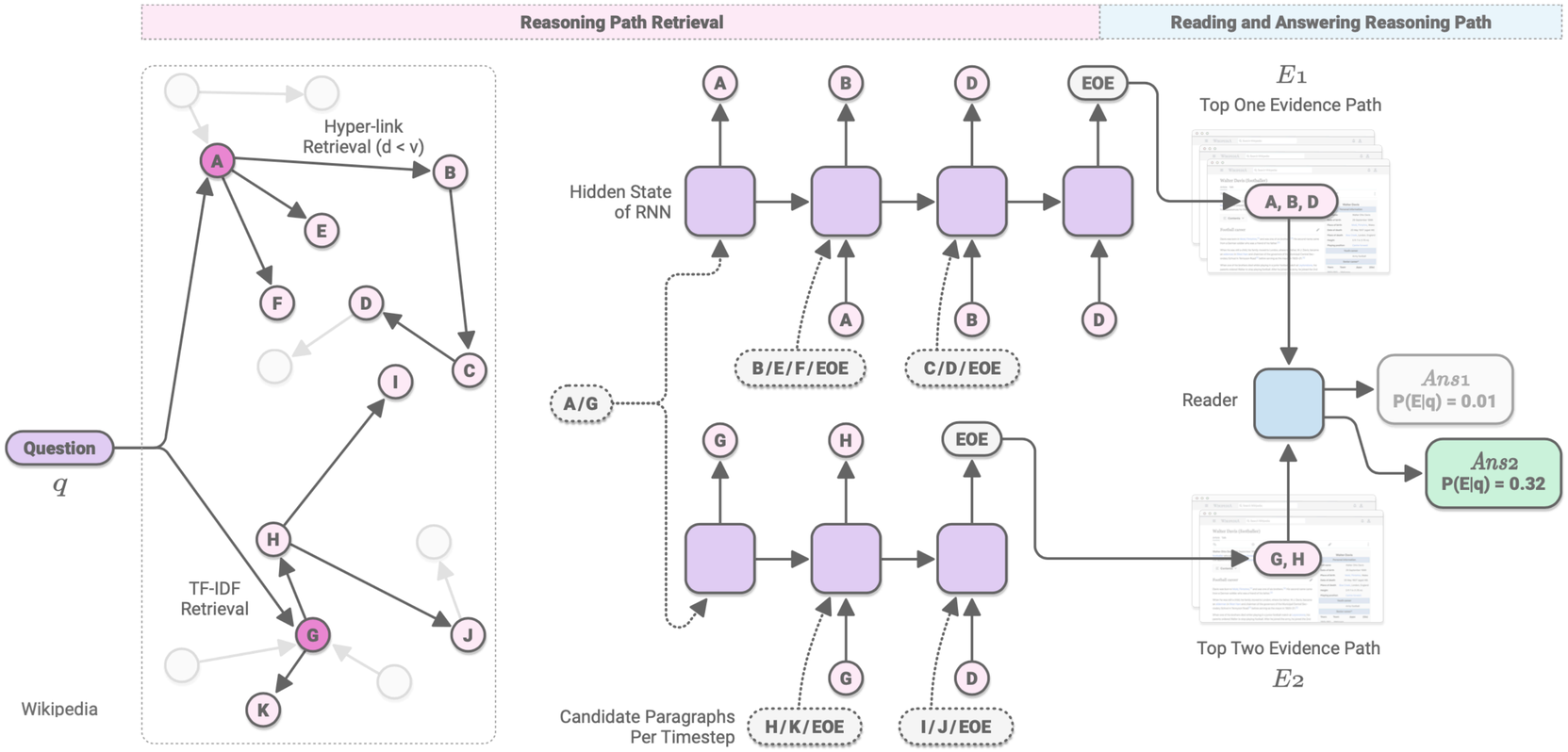
Akari Asai, Kazuma Hashimoto, Hannaneh Hajishirzi, Richard Socher, Caiming Xiong ICLR 2020 paper

Overall Framework

Retriever: learns to predict plausibility of the reasoning paths

Reader : jointly learns to predict the plausibility and answer the question

$$w_i = \mathrm{BERT}_{[\mathrm{CLS}]}(q, p_i) \in \mathbb{R}^d,$$
$$P(p_i|h_t) = \sigma(w_i \cdot h_t + b),$$
$$h_{t+1} = \mathrm{RNN}(h_t, w_i) \in \mathbb{R}^d,$$



Multi task learning: reading comprehension + path re-ranking

Akari Asai, Kazuma Hashimoto, Hannaneh Hajishirzi, Richard Socher, Caiming Xiong ICLR 2020 paper

Reasoning over a Virtual KB

Differentiable Reasoning over a Virtual Knowledge Base

- Some challenges:
  1) Complex questions requires mentions in **multiple documents**, which is expensive.
  2) Pipeline methods are inherently error-prone compared with end-to-end

- Contributions:
  1) Scaling multi-hop QA to large document collections

- Complex QA over a large text corpus that has been encoded in a query-independent manner

Main Framework

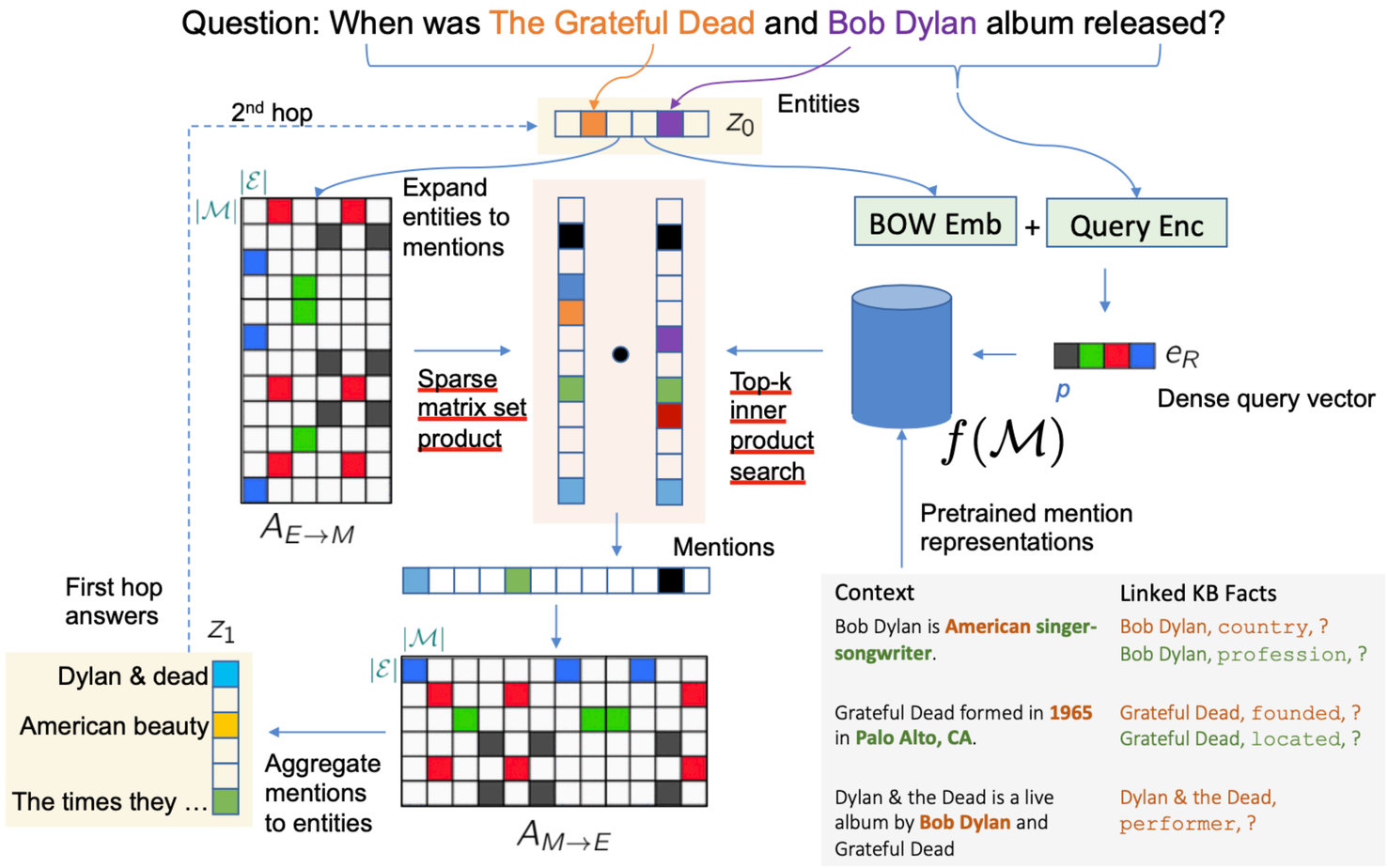


Figure 1: DrKIT answers multi-hop questions by iteratively mapping an input set of entities X (`The Grateful Dead`,`Bob Dylan`) to an output set of entities Y (`Dylan & Dead`, `American beauty`, ...) which are related to any input entity by some relation R (`album by`).

Differentiable Reasoning over a Virtual Knowledge Base 2020 ICLR full mark William W. Cohen etc.

Basic Ideas

- “multi-hop” complex queries which can be answered by **repeatedly executing a “soft” version of the operation**:

$$Y = X.\text{follow}(R) = \{x' : \exists x \in X \text{ s.t. } R(x, x') \text{ holds}\}$$

- Neural module to approximate this operation against an indexed corpus
- Treat a large corpus as a virtual KB by answering queries with spans from the corpus
- To simulate this behaviour on text, first expand z to set of co-occurring mentions (say using TF-IDF) m

Differentiable Reasoning over a Virtual Knowledge Base 2020 ICLR full mark William W. Cohen etc.

Basic Ideas

- Reasons about entity **mentions** $m = (e_m, k_m, i_m, j_m)$, denoting the text span $d_{k_m}^{i_m}, \ldots, d_{k_m}^{j_m}$ in document k_m mentions the entity e_m
- Probability of an intermediate answer: (recursively)

$$\Pr(z_t|q) = \sum_{z_{t-1} \in \mathcal{E}} \Pr(z_t|q, z_{t-1}) \Pr(z_{t-1}|q)$$

- $Pr(z_t \mid q, z_{t-1})$ needs to be aggregated over all mentions of z_t, thus rewrites as: $\Pr(z_t|q) = \sum_{m \in \mathcal{M}} \sum_{z_{t-1} \in \mathcal{E}} \Pr(z_t|m) \Pr(m|q, z_{t-1}) \Pr(z_{t-1}|q)$

(0-1 function indicating whether mention m matches the entity z)

- Joint modelling of **co-occurrence** and **relevance**:

A TF-IDF vector for the document containing m

$$\Pr(m|q, z_{t-1}) \propto \underbrace{\mathbb{1}\{G(z_{t-1}) \cdot F(m) > \epsilon\}}_{\text{expansion to co-occurring mentions}} \times \underbrace{s_t(m, z_{t-1}, q)}_{\text{relevance filtering}}$$

The TF-IDF vector of the *surface form* of the entity from the previous hop

A learnt scoring function (different for each hop)

Differentiable Reasoning over a Virtual Knowledge Base 2020 ICLR full mark William W. Cohen etc.

Basic Ideas

- Pre-compute the TFIDF term for all entities and mentions into a sparse matrix $A_{E \to M}[e, m] = \mathbb{1}\left(G(e) \cdot F(m) > \epsilon\right)$

$$\Pr(m|q, z_{t-1}) \propto \underbrace{\mathbb{1}\{G(z_{t-1}) \cdot F(m) > \epsilon\}}_{\text{expansion to co-occurring mentions}} \times \underbrace{s_t(m, z_{t-1}, q)}_{\text{relevance filtering}} \quad Z_t = [\Pr(z_t = e_1|q); \ldots; \Pr(z_t = e_{|\mathcal{E}|}|q)$$

The aggregation of mentions to entities: sparse matrix multiplication with coreference matrix

$$Z_t = \text{softmax}\left(\left[Z_{t-1}^T A_{E \to M} \odot \mathbb{T}_K(s_t(m, z_{t-1}, q))\right] A_{M \to E}\right).$$

(Top-k relevant mentions)

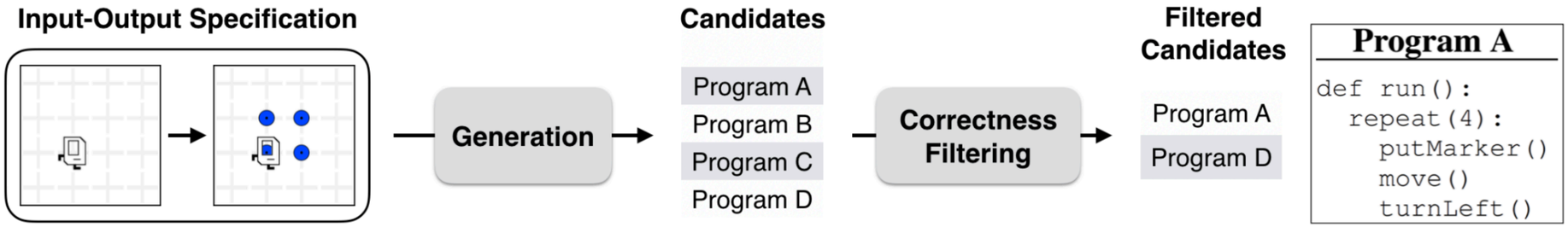
Sparse-matrix by sparse-vector multiplication = entity expansion to co-occuring mentions

Differentiable implementation of
$Z_t = Z_{t-1}.follow(R)$,
(i.e. mimicking the graph traversal in a traditional KB)

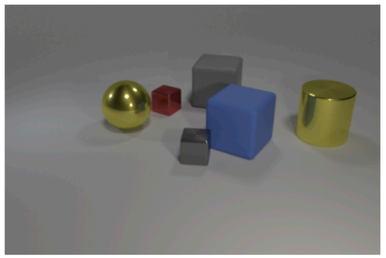
Reasoning via Program Induction

Basic ideas behind program induction for reasoning

- Motivation: human-like reasoning is **compositional**
- Explicitly **formulate and execute compositional plans** instead of learning a unexplainable mapping directly from inputs to outputs; (predicts a program representing the reasoning steps required to answer the question)
- Component: Program generator + Execution engine
- Advantages: explainable, direct reasoning to avoid dataset biases, ‘human-like’ compositional reasoning
- E.g.: Program Synthesis:



Showcases

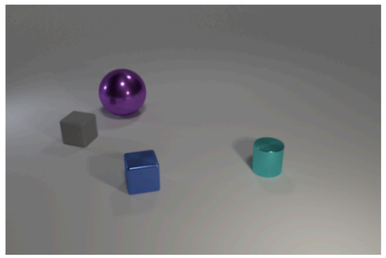


Q: *Is there a blue* box *in the* items*?* **A:** *yes*

Predicted Program:

```
exist
filter_shape[cube]
filter_color[blue]
scene
```

Predicted Answer:
✓ *yes*

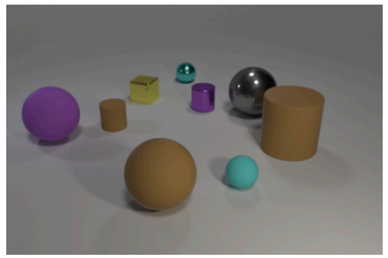


Q: *What shape object is* farthest *right?*
A: *cylinder*

Predicted Program:

```
query_shape
unique
relate[right]
unique
filter_shape[cylinder]
filter_color[blue]
scene
```

Predicted Answer:
✓ *cylinder*

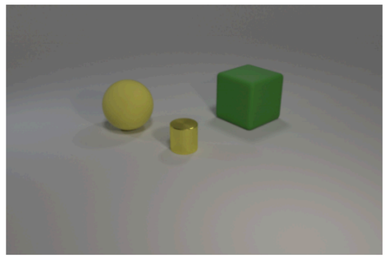


Q: *Are* all *the balls small?*
A: *no*

Predicted Program:

```
equal_size
query_size
unique
filter_shape[sphere]
scene
query_size
unique
filter_shape[sphere]
filter_size[small]
scene
```

Predicted Answer:
✓ *no*

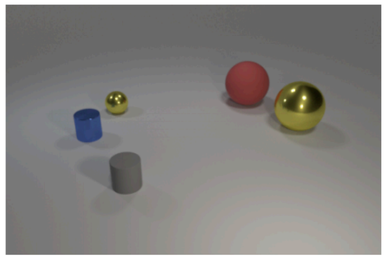


Q: *Is the green block to the right of the yellow sphere?*
A: *yes*

Predicted Program:

```
exist
filter_shape[cube]
filter_color[green]
relate[right]
unique
filter_shape[sphere]
filter_color[yellow]
scene
```

Predicted Answer:
✓*yes*



Q: Two items share *a color, a material, and a shape; what is the size of the* rightmost *of* those items*?* **A:** *large*

Predicted Program:

```
count
filter_shape[cube]
same_material
unique
filter_shape[cylinder]
scene
```

Predicted Answer:
✗ *0*

Johnson, Justin, Bharath Hariharan, Laurens Van Der Maaten, Judy Hoffman, Li Fei-Fei, C. Lawrence Zitnick, and Ross Girshick. "Inferring and executing programs for visual reasoning." ICCV 2017.

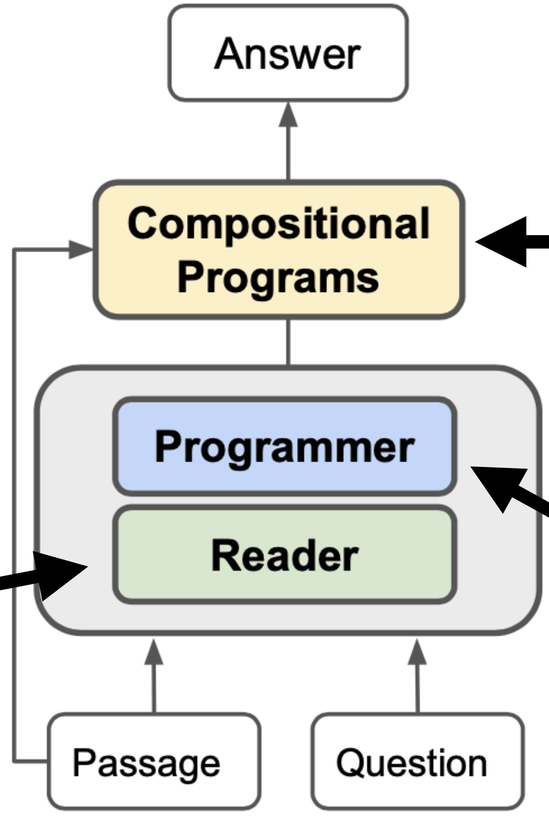
Challenges

- By adding ternary operators (if/then/else) and loops (for/do), the question “What color is the object with a unique shape?”
- Weak/Sparse supervision, i.e., with access only to the final answers (solutions: augmentation techniques and hard EM with thresholding)
- Needs to be domain-agnostic and compositional
- Operate over unstructured text (parsing the text into structured representations may introduce a lot of cascade errors)

Neural Symbolic Reader Overview

- Components:
  Reader, e.g., BERT, to encode the passage and question
  Programmer, e.g., LSTM, to generate a program that is executed to produce the answer.
- Basic ideas: introducing a set of **span selection operators**
  (PASSAGE_SPAN, QUESTION_SPAN, VALUE, KEY-VALUE)
- Collecting a set of programs for each question-answer pair

Neural Symbolic Reader Framework (General)



NeRd

Example:

Predicted Program:
COUNT(
PASSAGE_SPAN(75,75), PASSAGE_SPAN(77,78),
PASSAGE_SPAN(133,135), PASSAGE_SPAN(315,317))
Result: COUNT(‘38’,‘36 yards’, ‘43-yard’,‘52-yard’) = 4

Text: Passage + Question

Token embedding e’
Summary embedding e

Decodes a program as a sequence of tokens

Domain-specific language

Operator	Arguments	Outputs	Description
`PASSAGE_SPAN` `QUESTION_SPAN`	**v0**: the start index. **v1**: the end index.	a span.	Select a span from the passage or question.
`VALUE`	**v0**: an index.	a number.	Select a number from the passage.
`KEY-VALUE (KV)`	**v0**: a span. **v1**: a number.	a key-value pair.	Select a key (span) value (number) pair from the passage.
`DIFF` `SUM`	**v0**: a number or index. **v1**: a number or index.	a number.	Compute the difference or sum of two numbers.
`COUNT`	**v**: a set of spans.	a number.	Count the number of given spans.
`MAX` `MIN`	**v**: a set of numbers.	a number.	Select the maximum / minimum among the given numbers.
`ARGMAX` `ARGMIN`	**v**: a set of key-value pairs.	a span.	Select the key (span) with the highest / lowest value.

Table 1: Overview of our domain-specific language. See Table 2 for the sample usage.

Showcases

DROP

Passage	Question & Answer
Multiple spans	
...the population was spread out with **26.20% under the age of 18**, 9.30% from 18 to 24, **26.50% from 25 to 44**, **23.50% from 45 to 64**, and 14.60% who were 65 years of age or older...	**Question:** Which groups in percent are larger than 16%? **Program:** PASSAGE_SPAN(26,30), PASSAGE_SPAN(46,48), PASSAGE_SPAN(55,57) **Result:** 'under the age of 18', '25 to 44', '45 to 64'
Date	
When major general Nathanael Greene took command in the south, Marion and lieutenant colonel Henry Lee were ordered in January **1781**... On **August 31**, Marion rescued a small American force trapped by 500 British soldiers...	**Question:** When did Marion rescue the American force? **Program:** PASSAGE_SPAN(71,71), PASSAGE_SPAN(72,72), PASSAGE_SPAN(32,32) **Result:** 'August', '31', '1781'
Numerical operations	
...Lassen county had a population of **34,895**. The racial makeup of Lassen county was **25,532 (73.2%) white** (U.S. census), **2,834 (8.1%) African American** (U.S. census)...	**Question:** How many people were not either solely white or solely African American? **Program:** DIFF(SUM(9,10),12) **Result:** 34895 - 25532 - 2834 = 6529
Counting	
...the Bolshevik party came to power in November 1917 through the **simultaneous election in the soviets** and an **organized uprising supported by military mutiny**...	**Question:** How many factors were involved in bringing the Bolsheviks to power? **Program:** COUNT(PASSAGE_SPAN(62, 66), PASSAGE_SPAN(69, 74)) **Result:** COUNT('simultaneous election in the soviets', 'organized uprising supported by military mutiny') = 2

MathQA

Passage	Question & Answer
Multiple spans	
...the population was spread out with **26.20% under the age of 18**, 9.30% from 18 to 24, **26.50% from 25 to 44**, **23.50% from 45 to 64**, and 14.60% who were 65 years of age or older...	**Question:** Which groups in percent are larger than 16%? **Program:** PASSAGE_SPAN(26,30), PASSAGE_SPAN(46,48), PASSAGE_SPAN(55,57) **Result:** 'under the age of 18', '25 to 44', '45 to 64'
Date	
When major general Nathanael Greene took command in the south, Marion and lieutenant colonel Henry Lee were ordered in January **1781**... On **August 31**, Marion rescued a small American force trapped by 500 British soldiers...	**Question:** When did Marion rescue the American force? **Program:** PASSAGE_SPAN(71,71), PASSAGE_SPAN(72,72), PASSAGE_SPAN(32,32) **Result:** 'August', '31', '1781'
Numerical operations	
...Lassen county had a population of **34,895**. The racial makeup of Lassen county was **25,532 (73.2%) white** (U.S. census), **2,834 (8.1%) African American** (U.S. census)...	**Question:** How many people were not either solely white or solely African American? **Program:** DIFF(SUM(9,10),12) **Result:** 34895 - 25532 - 2834 = 6529
Counting	
...the Bolshevik party came to power in November 1917 through the **simultaneous election in the soviets** and an **organized uprising supported by military mutiny**...	**Question:** How many factors were involved in bringing the Bolsheviks to power? **Program:** COUNT(PASSAGE_SPAN(62, 66), PASSAGE_SPAN(69, 74)) **Result:** COUNT('simultaneous election in the soviets', 'organized uprising supported by military mutiny') = 2

Transformers as Soft Reasoners over Language

Transformers as Soft Reasoners over Language

- Goal: Reason over explicitly provided knowledge using rules expressed in natural language. (Treat Transformer as a limited **“soft theorem prover”** operating over explicit theories in language and test its ability to learn rule-based reasoning)
- Without storing, retrieving, or otherwise operating on structured symbolic expressions.
- Provides new possibilities for explainability, correctability, and counterfactual reasoning in question-answering

Transformers as Soft Reasoners over Language Peter Clark, Oyvind Tafjord, Kyle Richardson 2020

Showcases

Select an example:

Nails are made of iron, which is a metal, which conducts electricity. Ask: Nails conduct electricity.

Facts and rules (you can provide your own):

Metals conduct electricity.
Insulators do not conduct electricity.
If something is made of iron then it is metal.
Nails are made of iron.

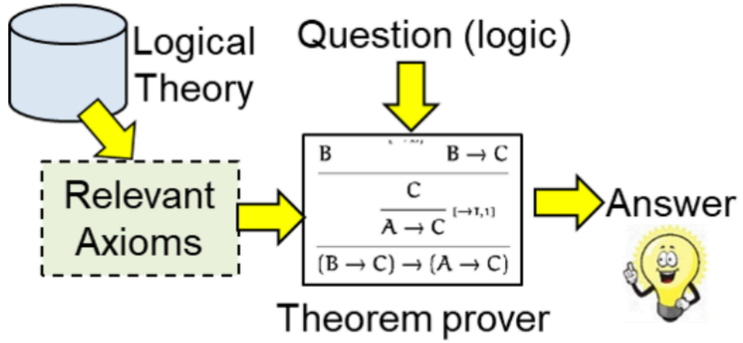
Is it true?

Nails conduct electricity.

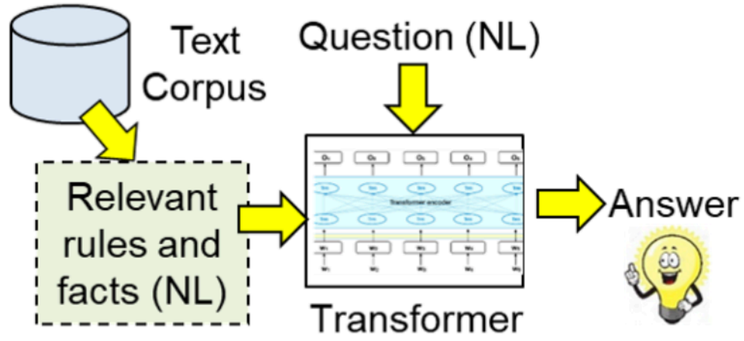
Submit

ROVER prediction:

- Nails conduct electricity. **True** (confidence = 0.99)



(a) formal reasoning



(b) linguistic analog

Training Details

- Data example: (context, statement, answer) := ((fact, rule), statement, answer)
  [Statement is the question, namely a declarative sentence to prove]
  [Answer is either T (true) if statement deductively follows from the context, or F if it does not (false under a closed-world assumption, CWA)]
- Train RoBERTa fro binary classification.
  Questions are supplied to RoBERTa as: [CLS] context [SEP] statement [SEP],
  context is the theory (facts+rules, expressed in language)
  statement is the fact to try and prove.
  The [CLS] output token is projected to a single logit.
- Evaluation in 5 datasets, each constrained by the maximum depth of inference required to prove the facts used in its questions (depth $\in (0,5)$)

Explanations Generation

Critical sentence predictions: **P:** 0.666667 **R:** 0.4 **F1:** 0.5
Statement: The squirrel is green. (TRUE) **Depth:** 3
Context: All rough things are young. The squirrel eats the tiger.
If something sees the squirrel and it is young then the squirrel is green.
If something eats the squirrel then it is rough. The tiger visits the bear. The tiger eats the squirrel.
The lion eats the tiger. If the squirrel sees the bear and the bear sees the tiger then the bear sees the lion.
Green things are rough. The lion eats the squirrel. If something is green then it is big. **The lion sees the squirrel.**
The tiger is young. If something visits the squirrel and it visits the bear then the squirrel visits the bear.
The tiger eats the bear. The bear is green. If something sees the squirrel then it eats the tiger. The tiger sees the bear.

Above, the model has identified two critical sentences (green), but has missed three (purple) and included an irrelevant rule (red). Here, the underlying line of reasoning is: "The lion eats the squirrel (fourth highlighted statement), therefore is rough (third), therefore young (first), and as it also sees the squirrel (sixth) it is green (second)."

Counterfactuals

C: *Metal things conduct electricity. Insulated things do not conduct electricity. Nails are iron.*

Q: *A nail conducts electricity?*

A: TRUE

C: *Metals conduct electricity. Insulators do not conduct electricity. Nails are insulators.*

Q: *A nail conducts electricity?*

A: FALSE

C: *Metals conduct electricity. Insulators do not conduct electricity. Plastic is a metal. Nails are plastic.*

Q: *A nail conducts electricity?*

A: TRUE

C: *Metals conduct electricity. Insulators do not conduct electricity. Plastic is an insulator. Nails are plastic.*

Q: *A nail conducts electricity?*

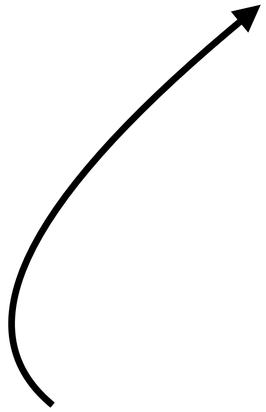
A: FALSE

Some weaknesses

- Only limited to trivial If-And-Then rules to conduct deductive reasoning, can’t induce new knowledge from training data
- Lack of commonsense knowledge:
- Can’t tolerate minor, unstated taxonomic gaps:
- Not robust to unseen relations:

Idea1: Transformer as logic rules inducer

- Learning to perform logical reasoning for **counterfactual text generation** with transformer
- Noise-tolerant: Probabilistic approach (BDL, Graphical Models etc)
- Now that transformers can be viewed as **“soft theorem prover”** over text **(logical deduction),** A natural question: Can transformers **induce new logic rules from raw text** (given background knowledge)?
  Or more general: lifelong learning with logical deduction and induction (learning with rich experience)
- Can we enhance the reasoning ability of transformers given logic rules via **knowledge distillation**?



CLUTRR: A Diagnostic Benchmark for Inductive Reasoning from Text.
Koustuv Sinha, Shagun Sodhani, Jin Dong, Joelle Pineau, William L. Hamilton EMNLP 2019

Idea2: Virtual KB

- Consider Two ways to reason over unstructured text:
  1. Phrase-indexed question answering 2. Program induction

- Most current **open-domain QA** methods focus on **relational questions**
  use a **pipeline approach** that includes a retriever and reader,
  while most **symbolic QA** methods are limited to **small scale** of texts,

  What about using phrase-indexed method to extend current **logical reasoning** to **large corpus** like traversing VKB? (e.g.: replace PASSAGE_SPAN operation)
  (either end-to-end blackbox framework or via end-to-end logic programs)

Some questions

- Task : focus on Multi-hop reasoning (general QA) or Single-hop reasoning (KG completion)?
- Setup: Weak(Sparse) supervision? (only know final answer)

THANK YOU