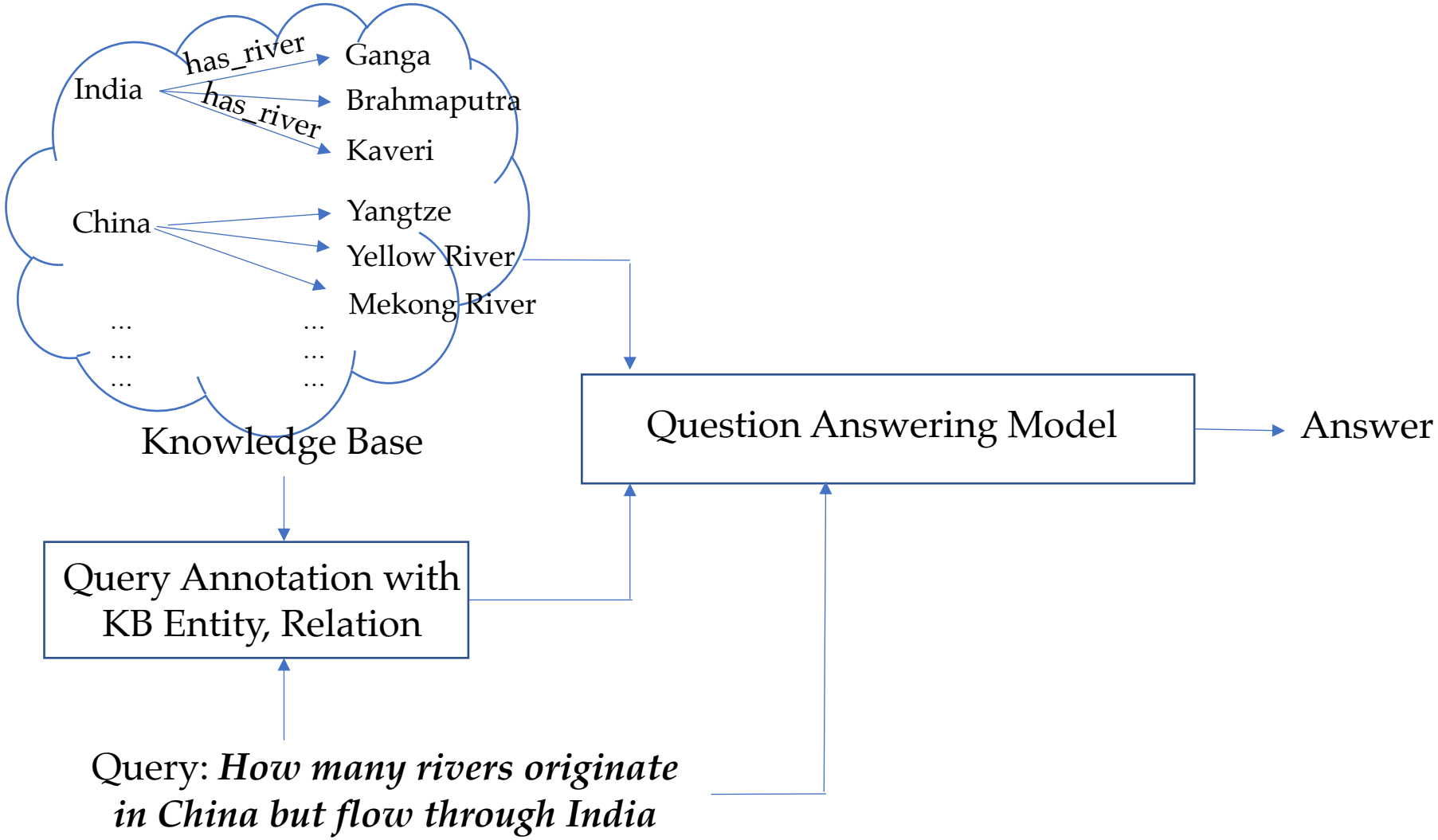
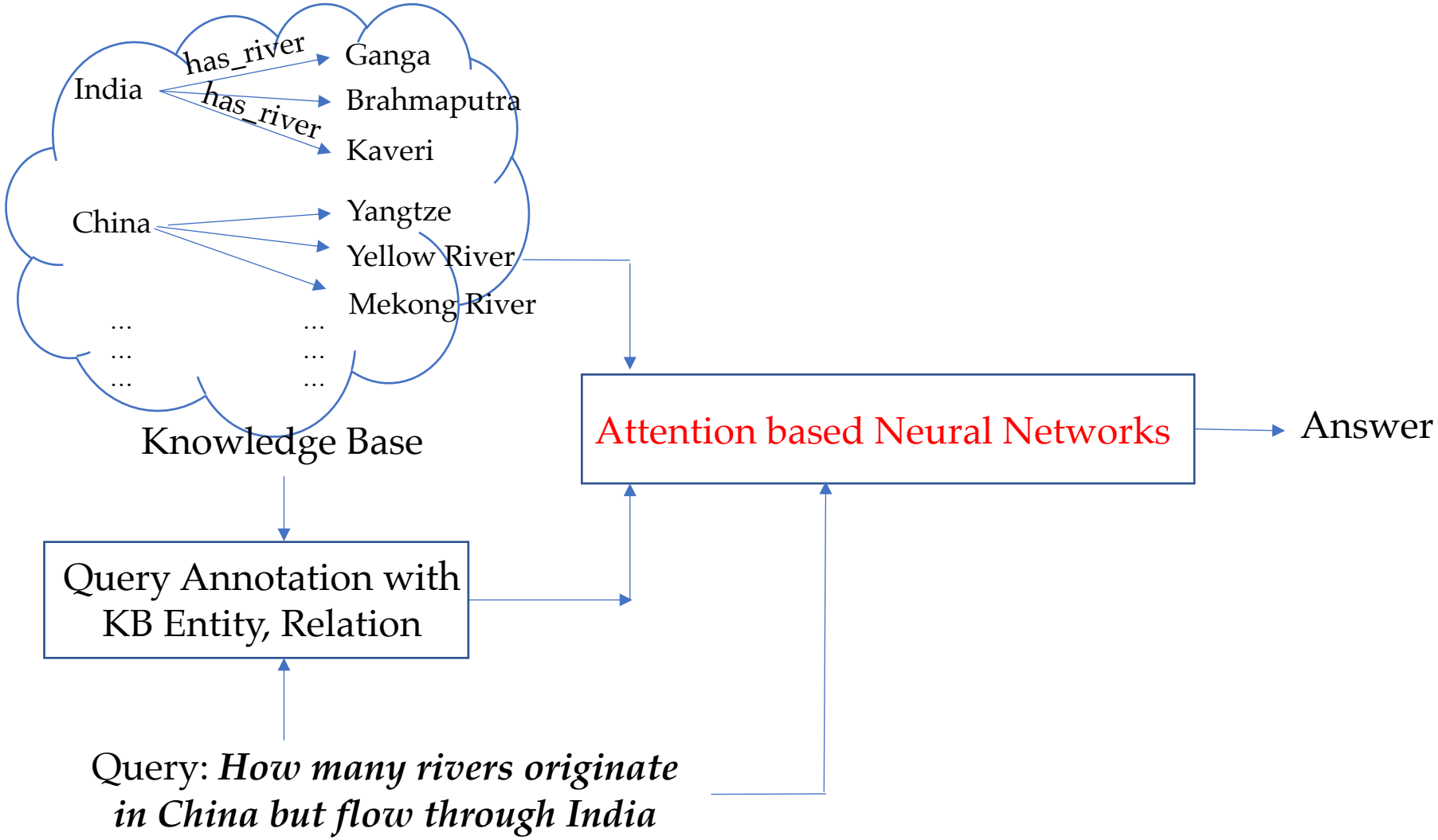
Complex Imperative Program Induction from Terminal Rewards

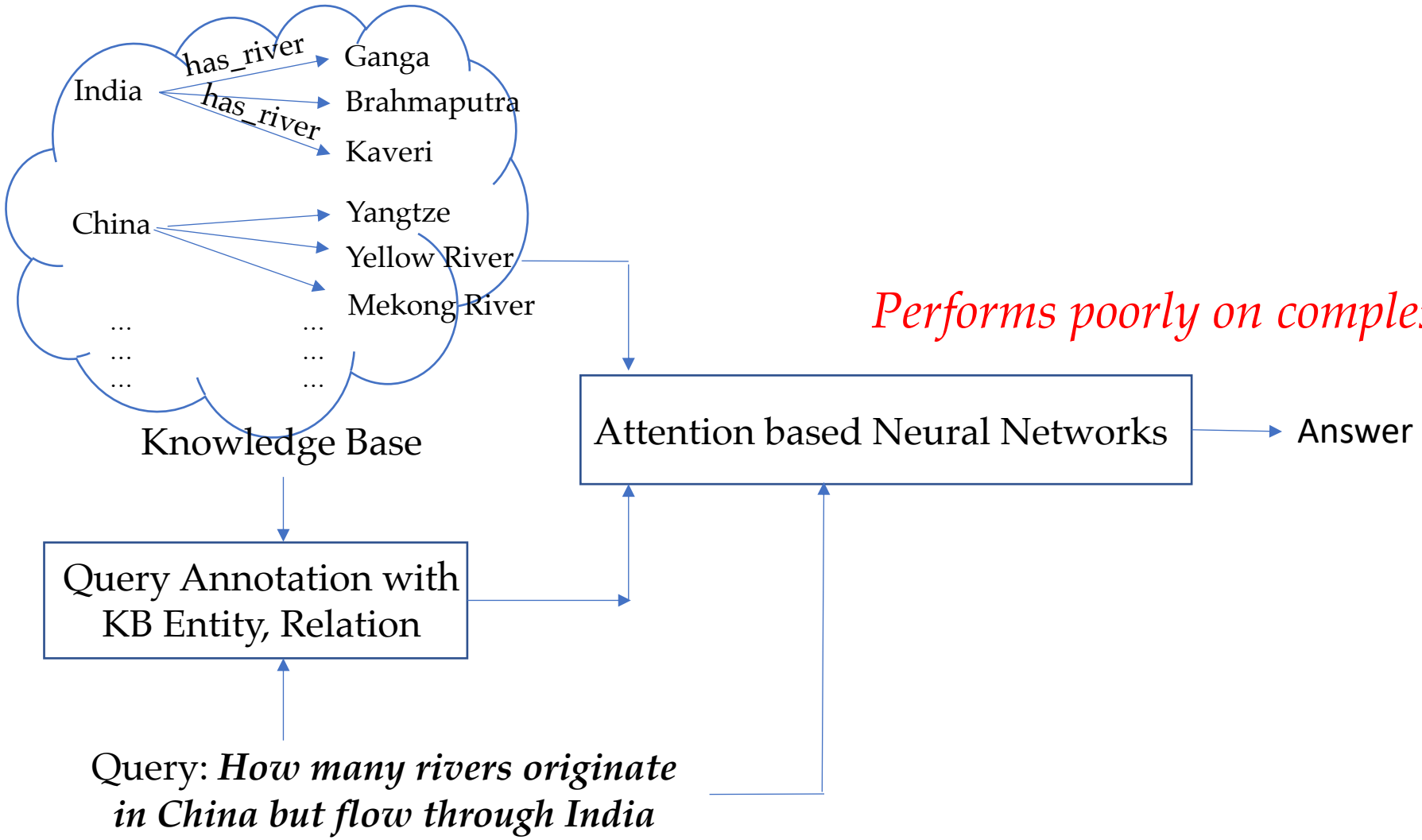
KB Based Question Answering (KBQA)



Option# 1. End-To-End Neural Models for KBQA

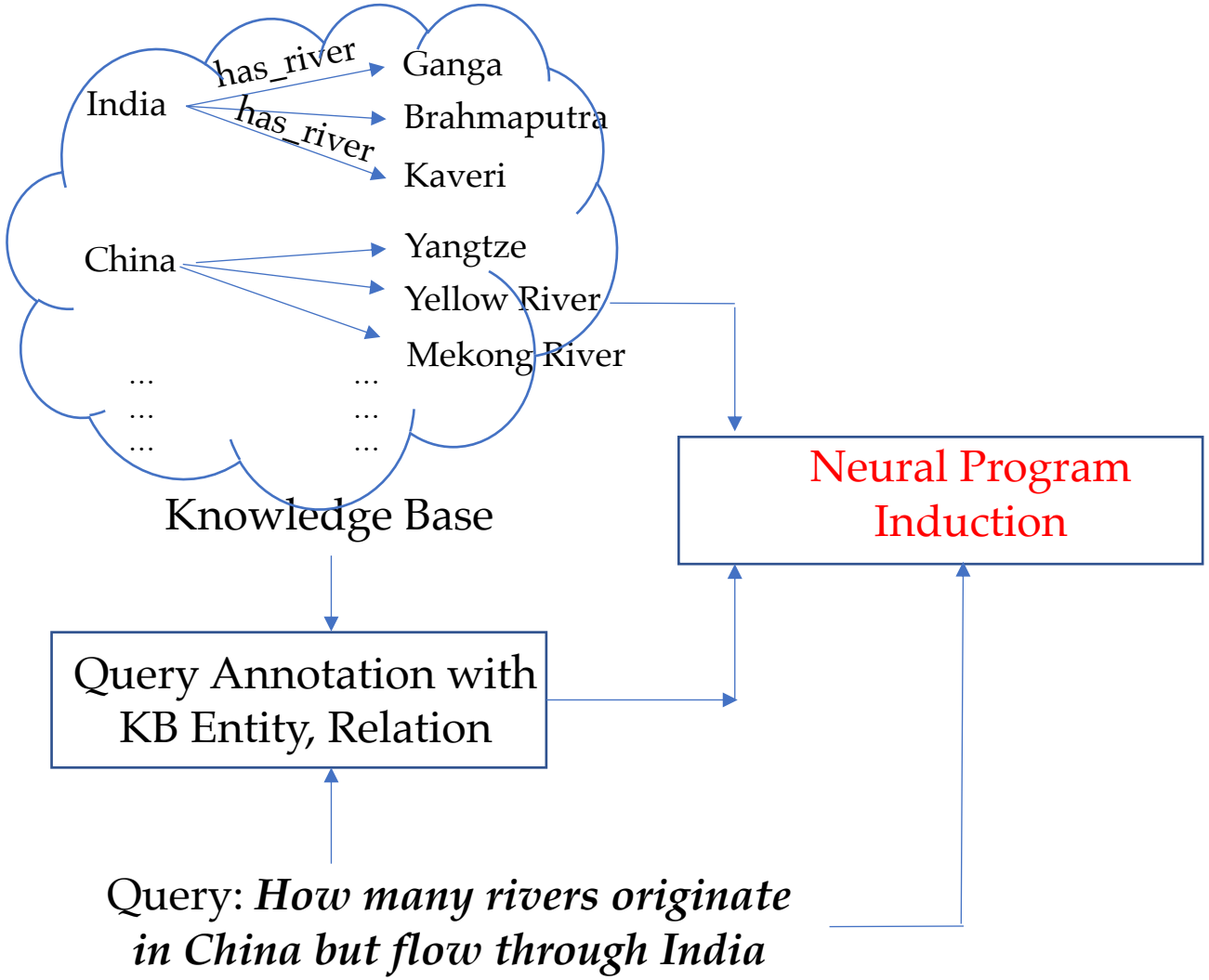


Option# 1. End-To-End Neural Models for KBQA

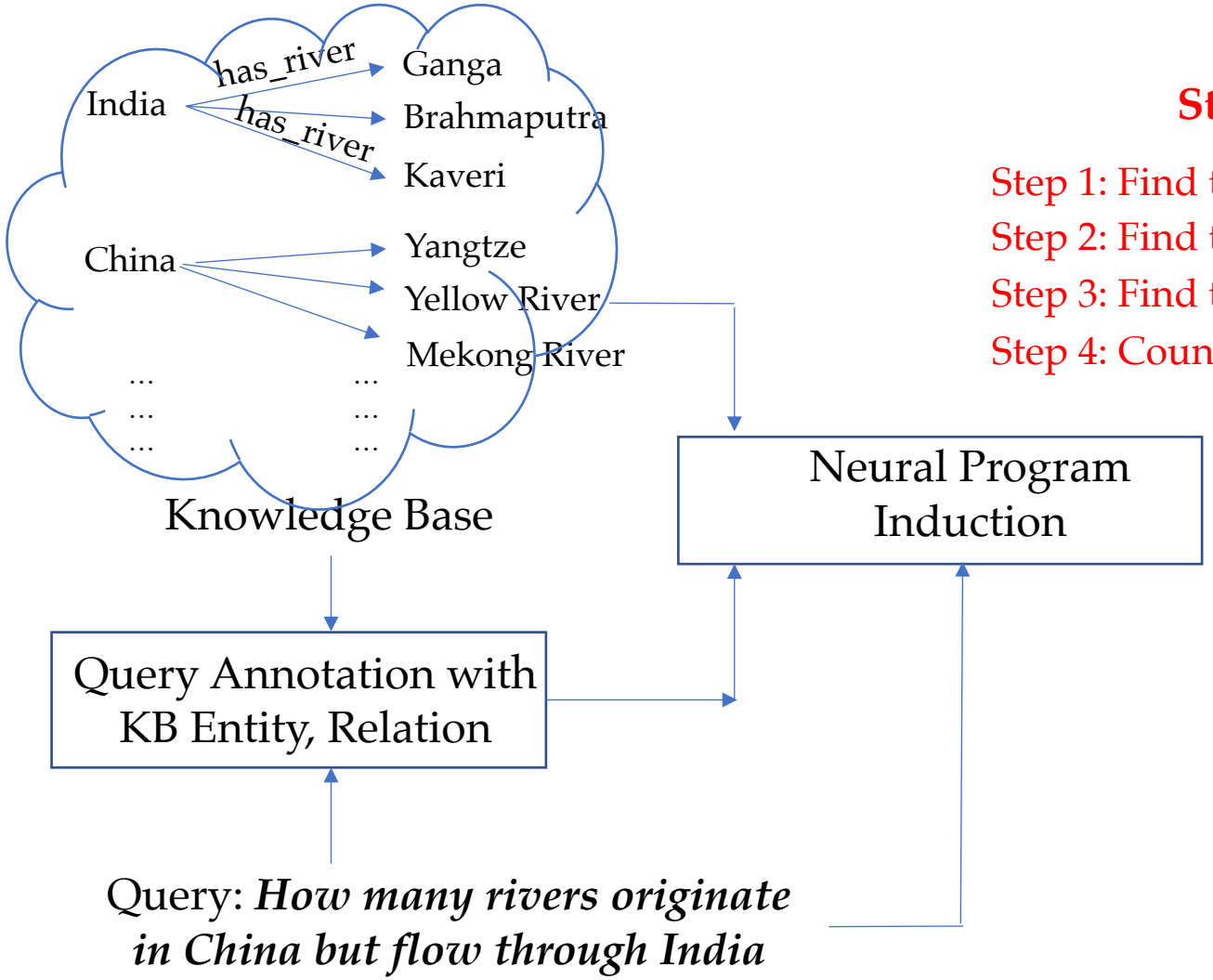


Performs poorly on complex QA!

Option# 2. Modular Neural Models for KBQA



Option# 2. Modular Style KBQA



Steps for Solving Query

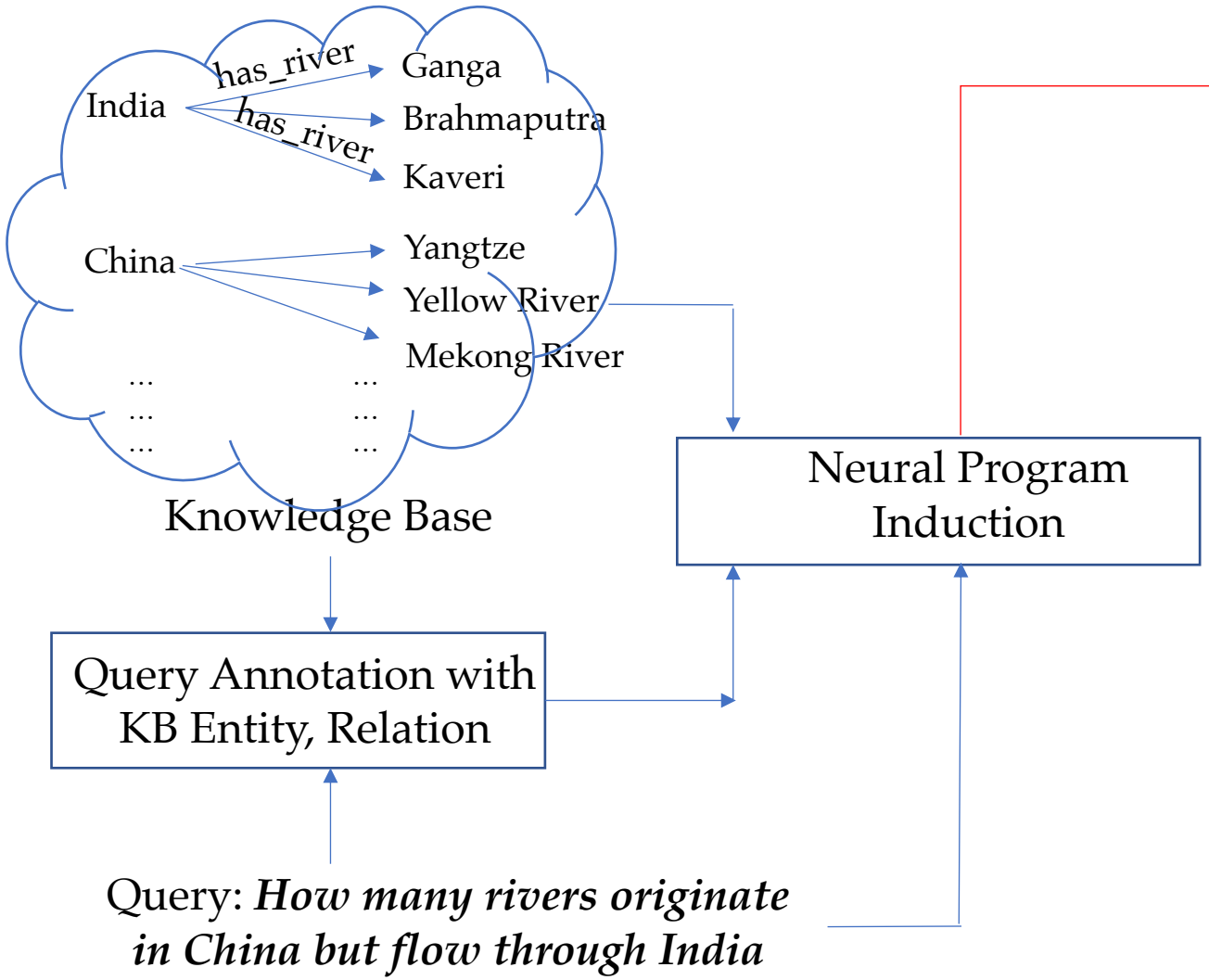
Step 1: Find the set of rivers that originate in China

Step 2: Find the set of rivers that flow through India

Step 3: Find the intersection of the two sets

Step 4: Count the number of rivers in the intersection

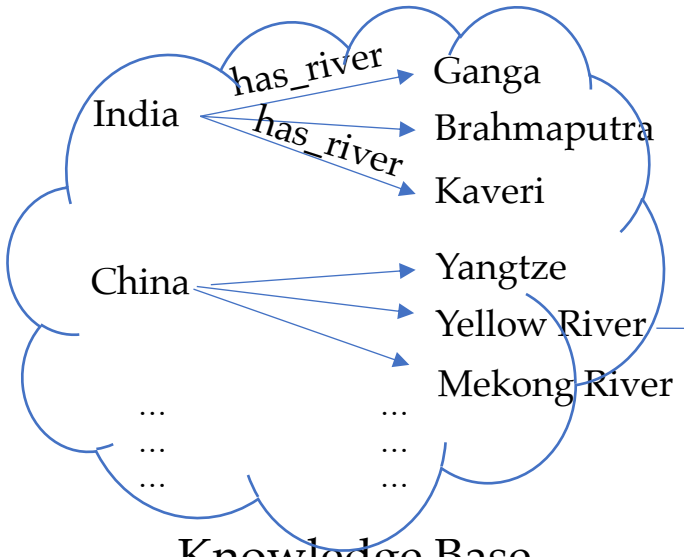
Program Induction Step



Generated Program

Step1: A = Generate_Set(China, has_origin, river)
Step2: B = Generate_Set(India, has, river)
Step3: C = Set_Intersection(A, B)
Step4: D = Count_Set_Size(C)

Program Execution Step



Knowledge Base

Query Annotation with KB Entity, Relation

Query: ***How many rivers originate in China but flow through India***

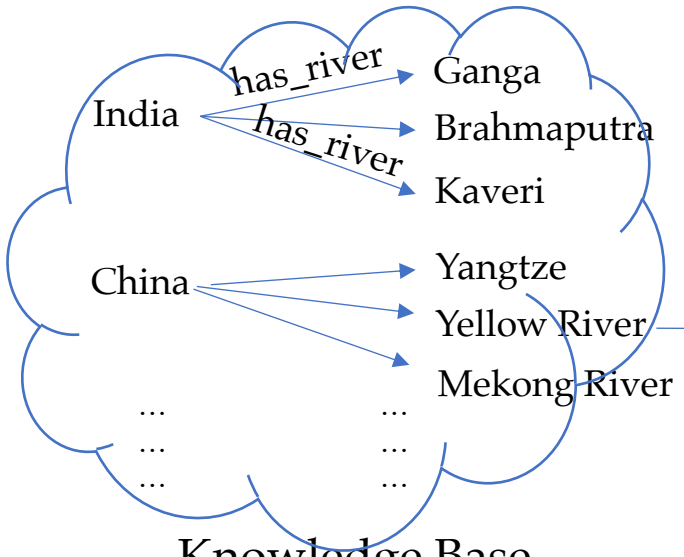
Neural Program Induction

Generated Program

Step1: A = Generate_Set(China, has_origin, river)
Step2: B = Generate_Set(India, has, river)
Step3: C = Set_Intersection(A, B)
Step4: D = Count_Set_Size(C)

Program Executor

Program Execution Step



Knowledge Base

Query Annotation with KB Entity, Relation

Query: ***How many rivers originate in China but flow through India***

Neural Program Induction

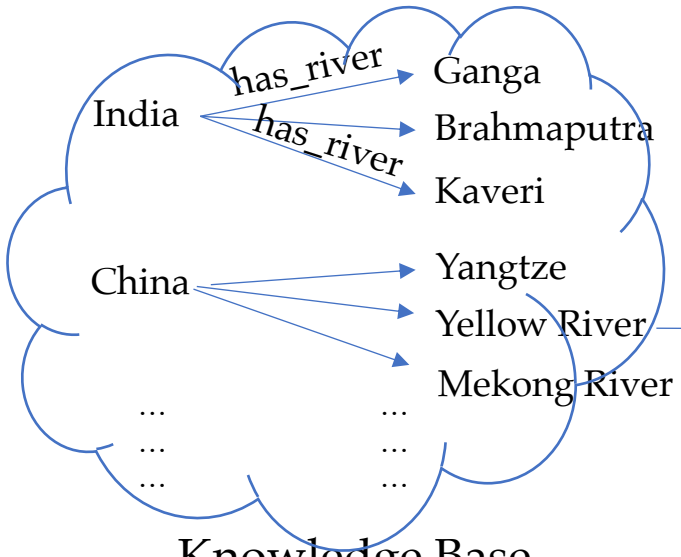
Generated Program

Step1: A = Generate_Set(China, has_origin, river)
Step2: B = Generate_Set(India, has, river)
Step3: C = Set_Intersection(A, B)
Step4: D = Count_Set_Size(C)

Program Executor

Step 1: A = { Yangtze, Yellow River, Mekong, Brahmaputra }
Step 2: B = { Brahmaputra, Ganga, Yamuna }
Step 3: C = { Brahmaputra }
Step 4: D = 1

Program Execution Step



Knowledge Base

Query Annotation with KB Entity, Relation

Query: ***How many rivers originate in China but flow through India***

Neural Program Induction

Generated Program

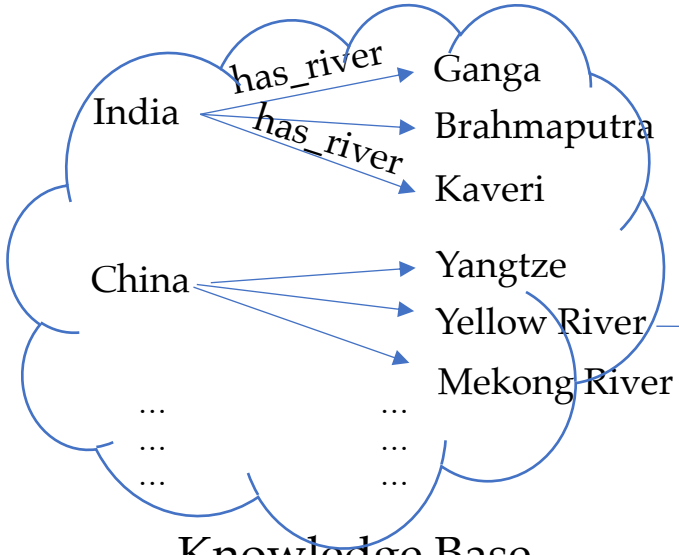
Step1: A = Generate_Set(China, has_origin, river)
Step2: B = Generate_Set(India, has, river)
Step3: C = Set_Intersection(A, B)
Step4: D = Count_Set_Size(C)

Program Executor

Step 1: A = { Yangtze, Yellow River, Mekong, Brahmaputra }
Step 2: B = { Brahmaputra, Ganga, Yamuna }
Step 3: C = { Brahmaputra }
Step 4: D = 1

Output = 1

Weak Supervised Program Induction for KBQA



Knowledge Base

Query Annotation with KB Entity, Relation

Query: ***How many rivers originate in China but flow through India***

Neural Program Induction

Getting Gold Program as supervision is hard!

Generated Program

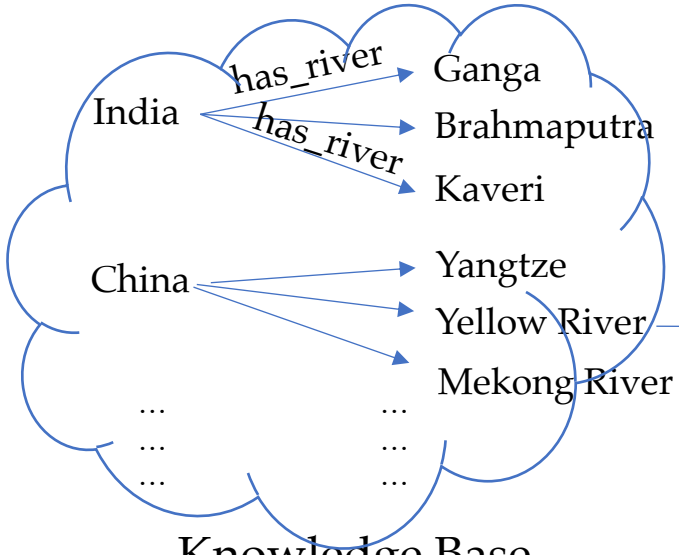
Step1: A = Generate_Set(China, has_origin, river)
Step2: B = Generate_Set(India, has, river)
Step3: C = Set_Intersection(A, B)
Step4: D = Count_Set_Size(C)

Program Executor

Step 1: A = { Yangtze, Yellow River, Mekong, Brahmaputra }
Step 2: B = { Brahmaputra, Ganga, Yamuna }
Step 3: C = { Brahmaputra }
Step 4: D = 1

Output = 1

Weak Supervised Program Induction for KBQA



Knowledge Base

Query Annotation with KB Entity, Relation

Query: ***How many rivers originate in China but flow through India***

Neural Program Induction

Generated Program

Step1: A = Generate_Set(China, has_origin, river)
Step2: B = Generate_Set(India, has, river)
Step3: C = Set_Intersection(A, B)
Step4: D = Count_Set_Size(C)

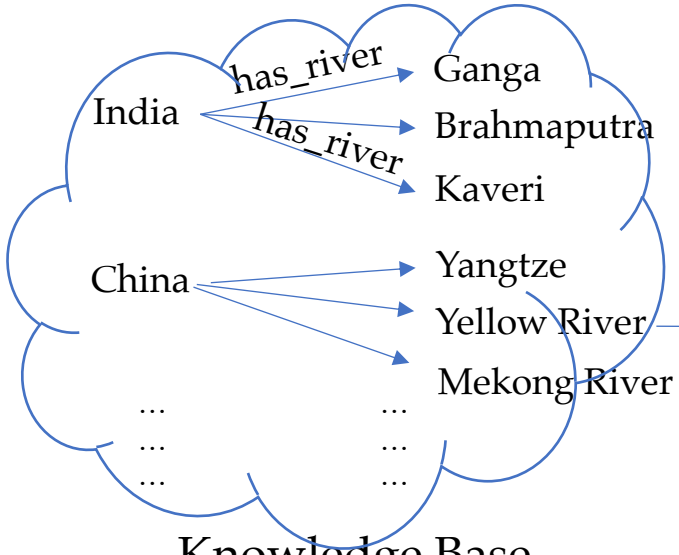
Program Executor

Step 1: A = { Yangtze, Yellow River, Mekong, Brahmaputra }
Step 2: B = { Brahmaputra, Ganga, Yamuna }
Step 3: C = { Brahmaputra }
Step 4: D = 1

Output = 1

But QA pairs are easily available

Weak Supervised Program Induction for KBQA



Knowledge Base

Query Annotation with KB Entity, Relation

Query: ***How many rivers originate in China but flow through India***

Neural Program Induction

Generated Program

Step1: A = Generate_Set(China, has_origin, river)
Step2: B = Generate_Set(India, has, river)
Step3: C = Set_Intersection(A, B)
Step4: D = Count_Set_Size(C)

Program Executor

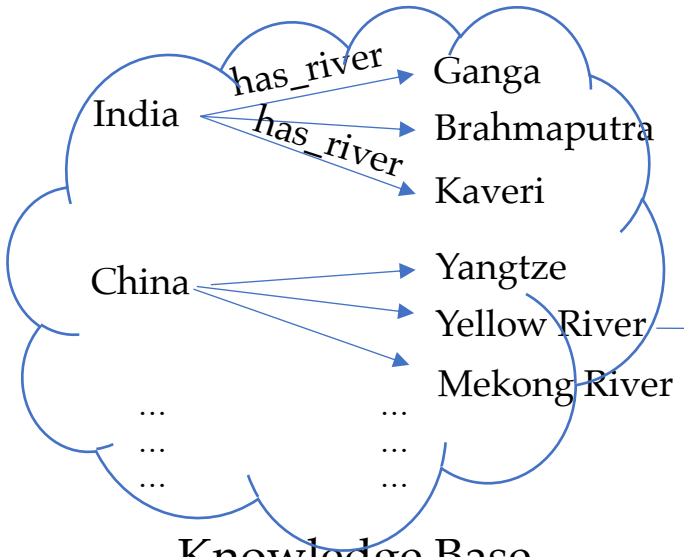
Step 1: A = { Yangtze, Yellow River, Mekong, Brahmaputra }
Step 2: B = { Brahmaputra, Ganga, Yamuna }
Step 3: C = { Brahmaputra }
Step 4: D = 1

Output = 1

Update Model based on Reward

CIPITR for KBQA

Complex Imperative Program Induction from Terminal Rewards (CIPITR)



Knowledge Base

Query Annotation with KB Entity, Relation

Query: ***How many rivers originate in China but flow through India***

Neural Program Induction

Generated Program

Step1: A = Generate_Set(China, has_origin, river)
Step2: B = Generate_Set(India, has, river)
Step3: C = Set_Intersection(A, B)
Step4: D = Count_Set_Size(C)

Program Executor

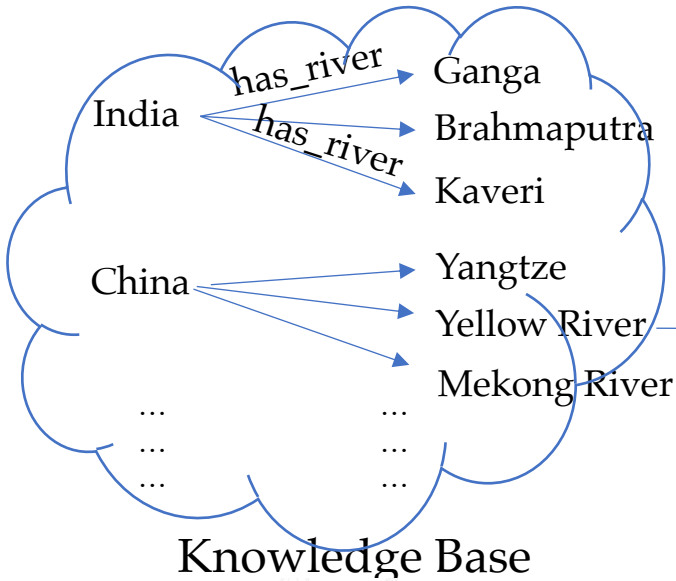
Update Model based on Reward

Step 1: A = { Yangtze, Yellow River, Mekong, Brahmaputra }
Step 2: B = { Brahmaputra, Ganga, Yamuna }
Step 3: C = { Brahmaputra }
Step 4: D = 1

Output = 1

CIPITR for KBQA

Complex Imperative Program Induction from Terminal Rewards (CIPITR)



Knowledge Base

Query A ... on with KB E... ation

Use Oracle Query Annotation

Neural Program Induction

Generated Program

Step1: A = Generate_Set(China, has_origin, river)
Step2: B = Generate_Set(India, has, river)
Step3: C = Set_Intersection(A, B)
Step4: D = Count_Set_Size(C)

Program Executor

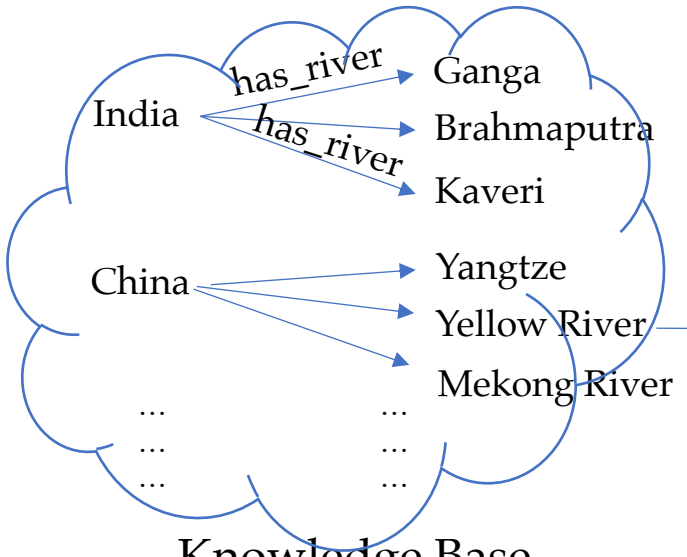
Update Model based on Reward

Step 1: A = { Yangtze, Yellow River, Mekong, Brahmaputra }
Step 2: B = { Brahmaputra, Ganga, Yamuna }
Step 3: C = { Brahmaputra }
Step 4: D = 1

Output = 1

s originate in China but flow through India

CIPITR for KBQA: Challenges



Knowledge Base

Query Annotation with KB Entity, Relation

Query: ***How many rivers originate in China but flow through India***

Neural Program Induction

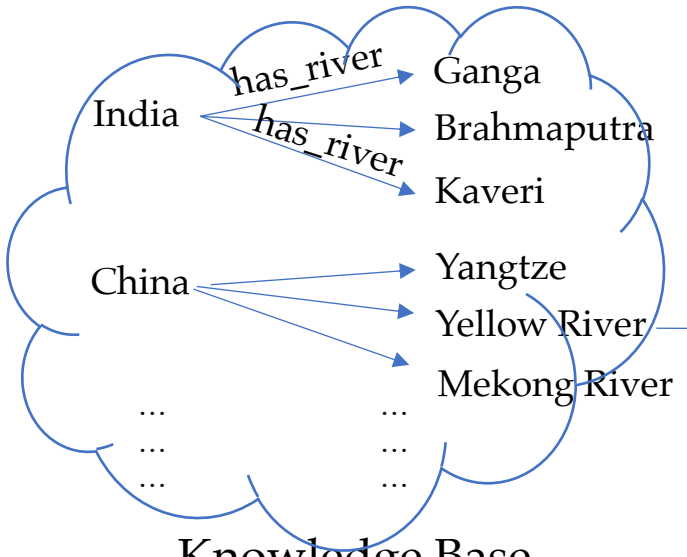
Update Model based on Reward

Challenge# 1: Long Programs owing to complex questions

Challenge# 2: Exponential Search Space of Programs

Challenge# 3: Extreme Reward Sparsity

CIPITR for KBQA: Challenges



Knowledge Base

Query Annotation with KB Entity, Relation

Query: ***How many rivers originate in China but flow through India***

Neural Program Induction

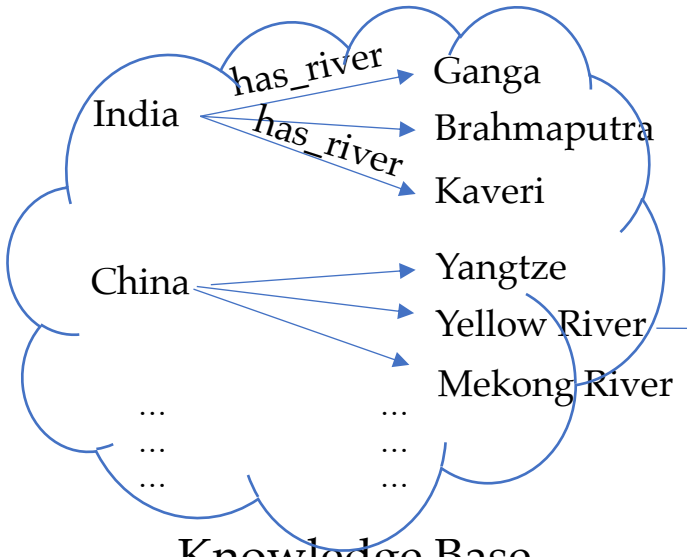
Update Model based on Reward

Challenge# 1: Long Programs owing to complex questions

Challenge# 2: Exponential Search Space of Programs

Challenge# 3: Extreme Reward Sparsity

CIPITR for KBQA: Challenges



Knowledge Base

Query Annotation with KB Entity, Relation

Query: ***How many rivers originate in China but flow through India***

Neural Program Induction

Update Model based on Reward

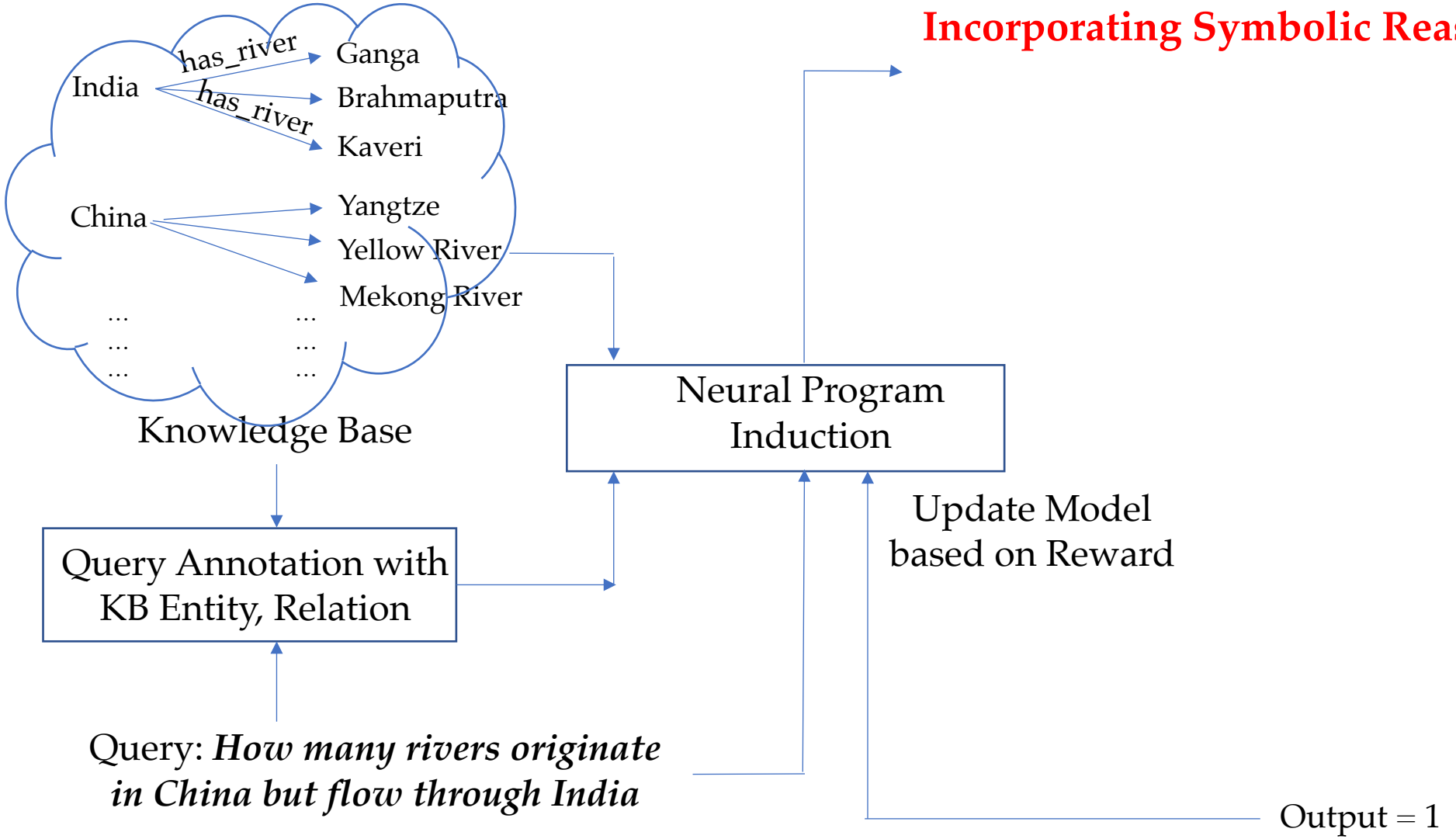
Challenge# 1: Long Programs owing to complex questions

Challenge# 2: Exponential Search Space of Programs

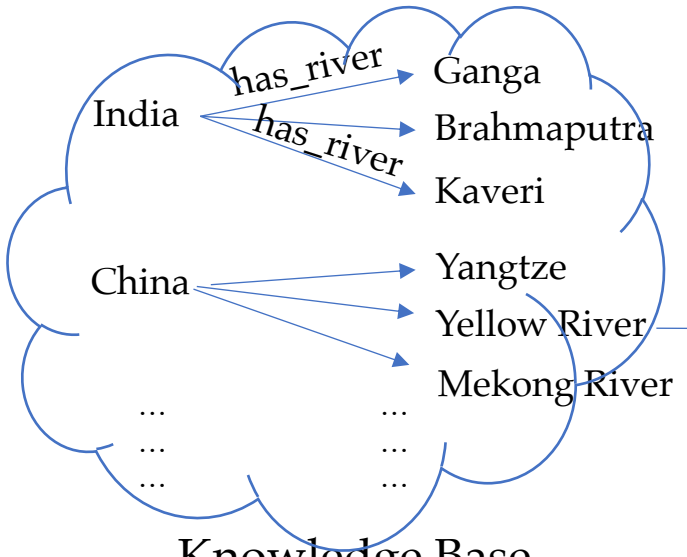
Challenge# 3: Extreme Reward Sparsity

Addressing the challenges by …

Incorporating Symbolic Reasoning in NPI



Addressing the challenges by …



Knowledge Base

Query Annotation with KB Entity, Relation

Query: ***How many rivers originate in China but flow through India***

Neural Program Induction

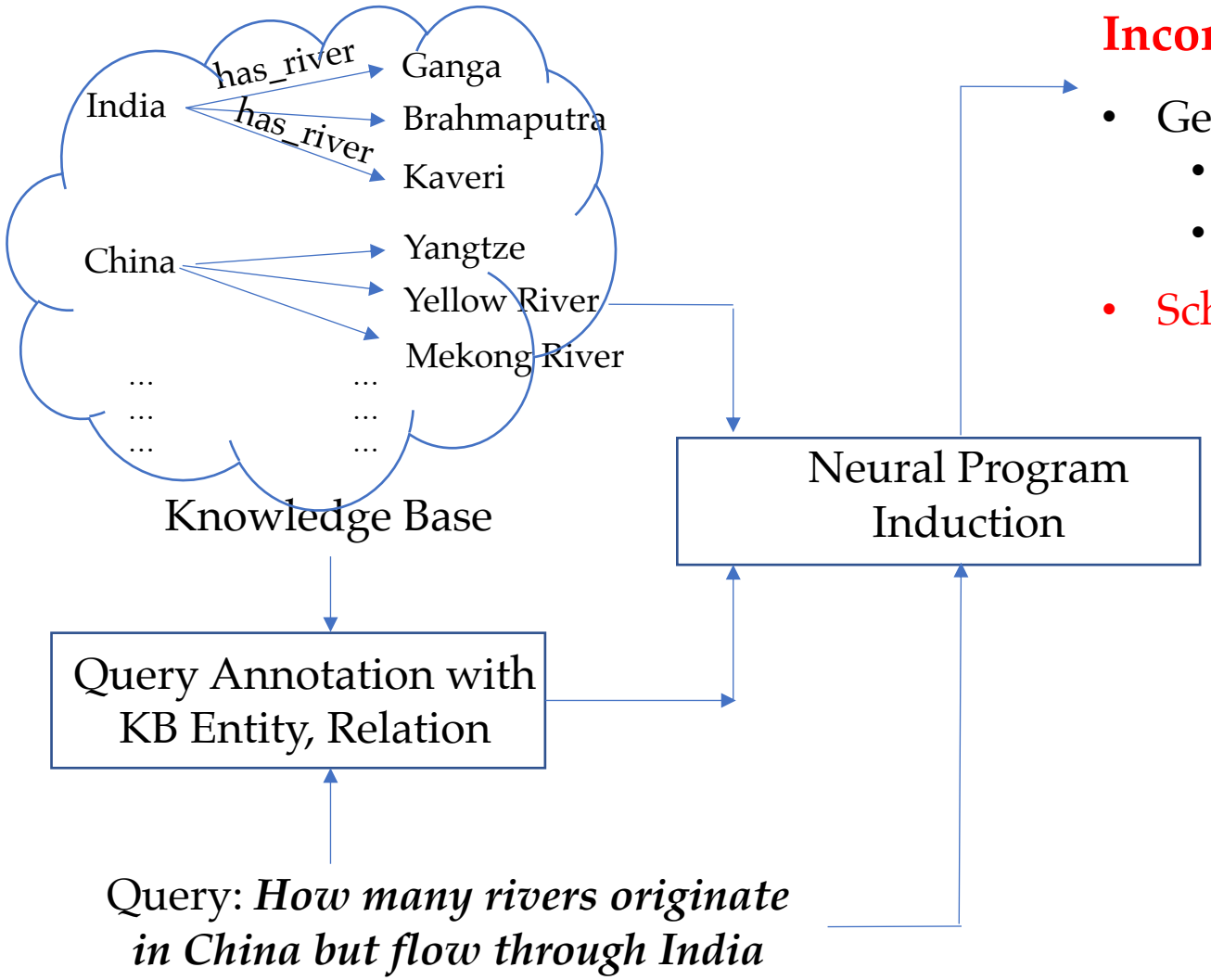
Update Model based on Reward

Output = 1

Incorporating Symbolic Reasoning in NPI

- Generic Programmatic Styles
  - No repetition of lines of program
  - Decomposition of program generation into phases

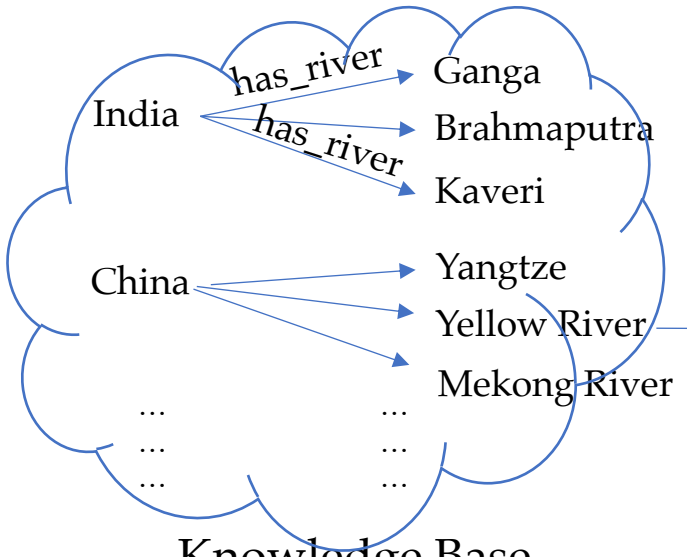
Addressing the challenges by …



Incorporating Symbolic Reasoning in NPI

- Generic Programmatic Styles
  - No repetition of lines of program
  - Decomposition of program generation into phases
- Schema and Task(KB) Specific constraints

Addressing the challenges by …



Neural Program Induction

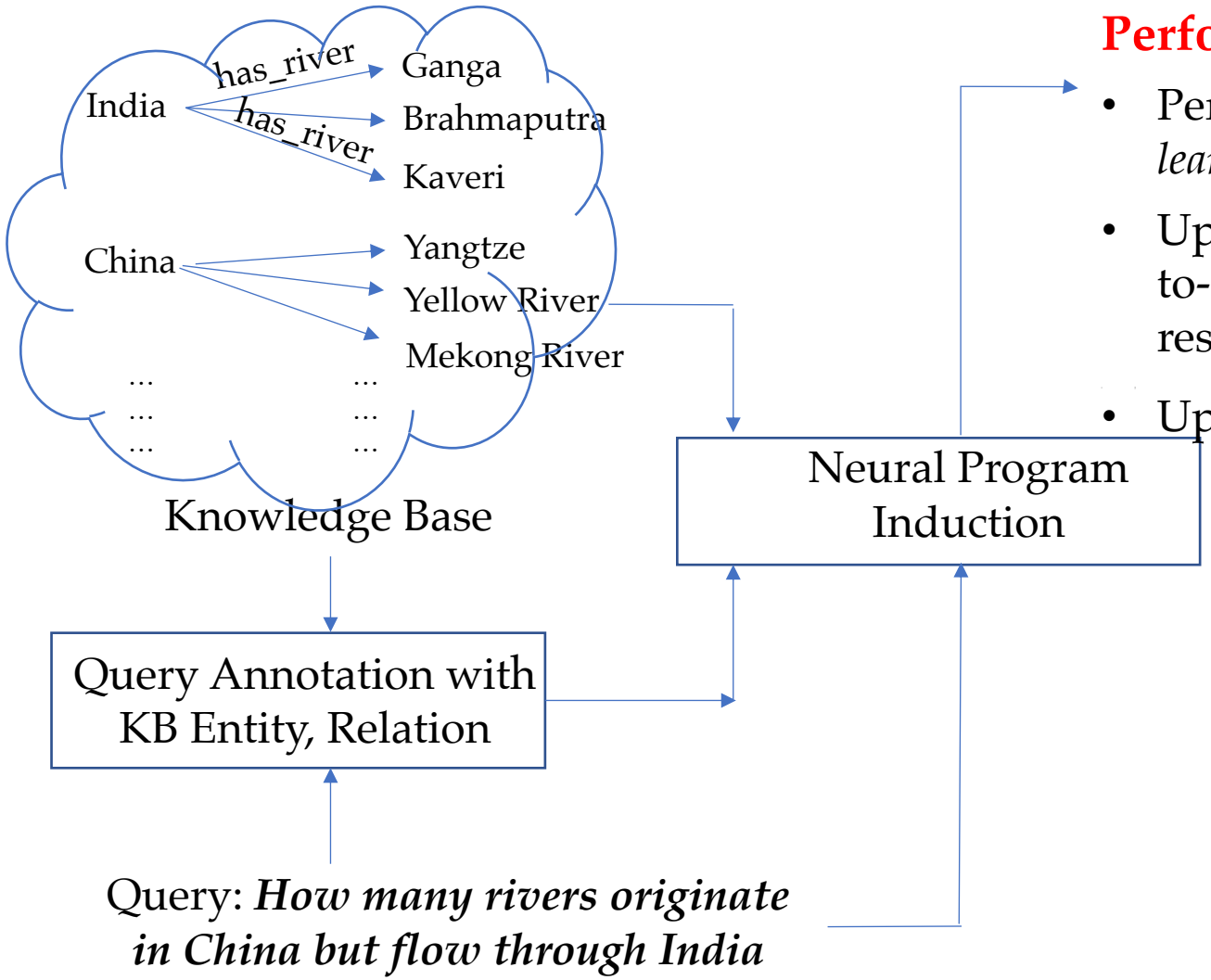
Query Annotation with KB Entity, Relation

Query: ***How many rivers originate in China but flow through India***

Incorporating Symbolic Reasoning in NPI

- Generic Programmatic Styles
  - No repetition of lines of program
  - Decomposition of program generation into phases
- Schema and Task(KB) Specific constraints
- Semantically Correct Programs
  - Biasing towards answer of meaningful type
  - Curriculum Learning of Answer Type first, value later

Performance Comparison …



Performance comparison with existing models

- Performs slightly better than RuleBased model despite *learning* rules
- Upto *20*× & *89*× improvement over existing NPI and End-to-End Neural models for 2-5 step & 5-10 step programs resp.
- Upto *5*× better generalizability over 7 different query types