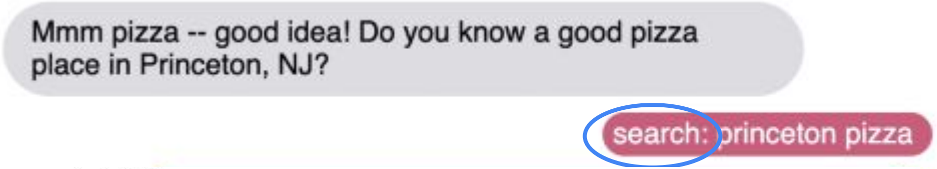
Carnegie Mellon University
Language Technologies Institute

TroVE: Inducing Verifiable and Efficient Toolboxes for Solving Programmatic Tasks

Zhiruo Wang[1] **Graham Neubig**[1] **Daniel Fried**[1]

What are tools?

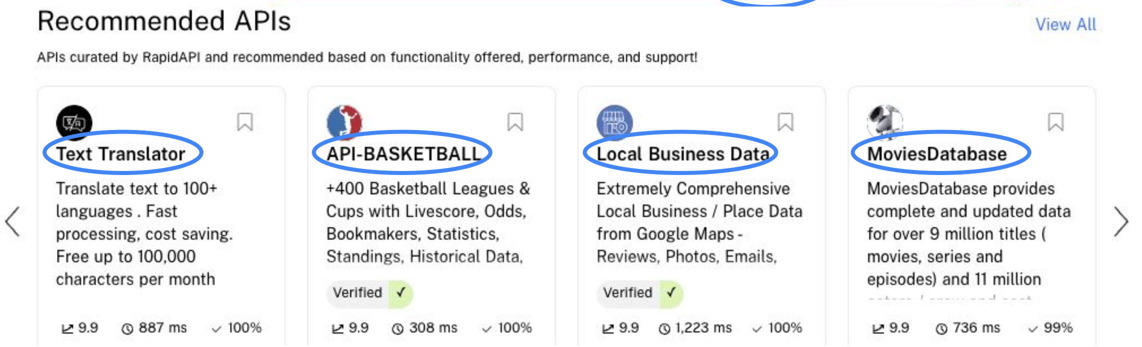
Application-specific software



Mmm pizza -- good idea! Do you know a good pizza place in Princeton, NJ?

search: princeton pizza

Real-world APIs



Recommended APIs

View All

APIs curated by RapidAPI and recommended based on functionality offered, performance, and support!

Text Translator

Translate text to 100+ languages . Fast processing, cost saving. Free up to 100,000 characters per month

9.9 887 ms 100%

API-BASKETBALL

+400 Basketball Leagues & Cups with Livescore, Odds, Bookmakers, Statistics, Standings, Historical Data,

Verified ✓

9.9 308 ms 100%

Local Business Data

Extremely Comprehensive Local Business / Place Data from Google Maps - Reviews, Photos, Emails,

Verified ✓

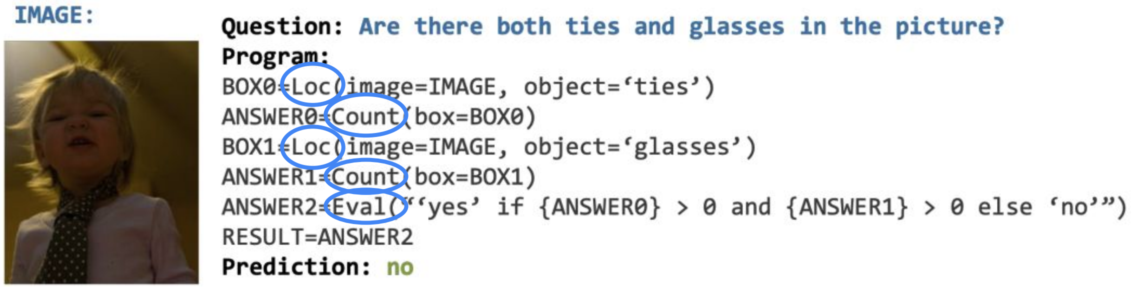
9.9 1,223 ms 100%

MoviesDatabase

MoviesDatabase provides complete and updated data for over 9 million titles (movies, series and episodes) and 11 million

9.9 736 ms 99%

Expert-designed functions

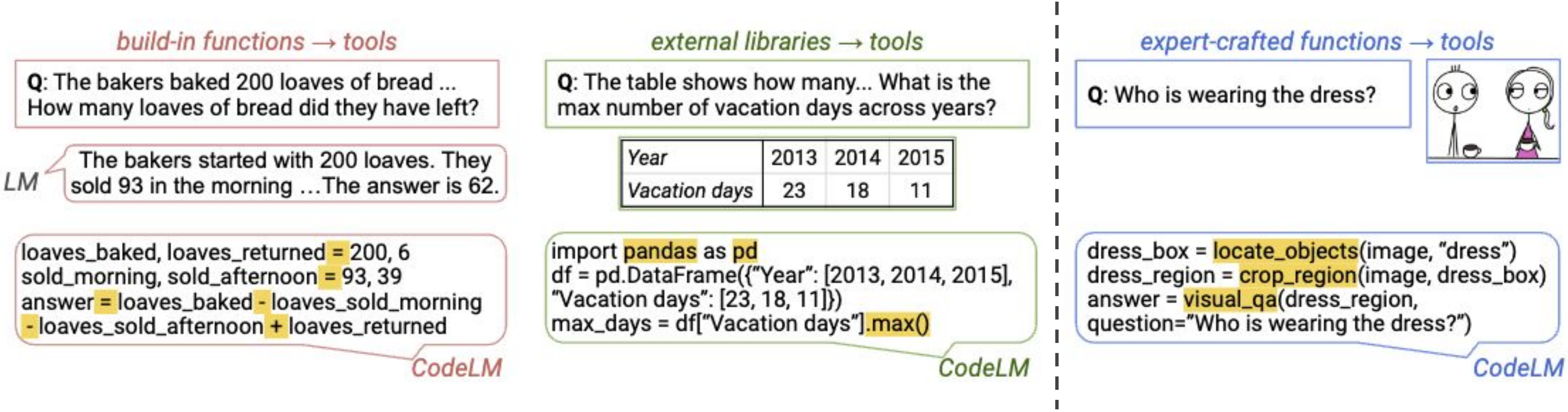


```
IMAGE:
Question: Are there both ties and glasses in the picture?
Program:
BOX0=Loc(image=IMAGE, object='ties')
ANSWER0=Count(box=BOX0)
BOX1=Loc(image=IMAGE, object='glasses')
ANSWER1=Count(box=BOX1)
ANSWER2=Eval("'yes' if {ANSWER0} > 0 and {ANSWER1} > 0 else 'no'")
RESULT=ANSWER2
Prediction: no
```

Can we make tools to improve task performance?

- On programmatic tasks, yes!

TROVE: Inducing Verifiable and Efficient Toolboxes for Solving Programmatic Tasks



Q: The bakers baked 200 loaves of bread ... How many loaves of bread did they have left?

The bakers started with 200 loaves. They sold 93 in the morning …The answer is 62.

```
loaves_baked, loaves_returned = 200, 6
sold_morning, sold_afternoon = 93, 39
answer = loaves_baked - loaves_sold_morning
- loaves_sold_afternoon + loaves_returned
```

Q: The table shows how many... What is the max number of vacation days across years?

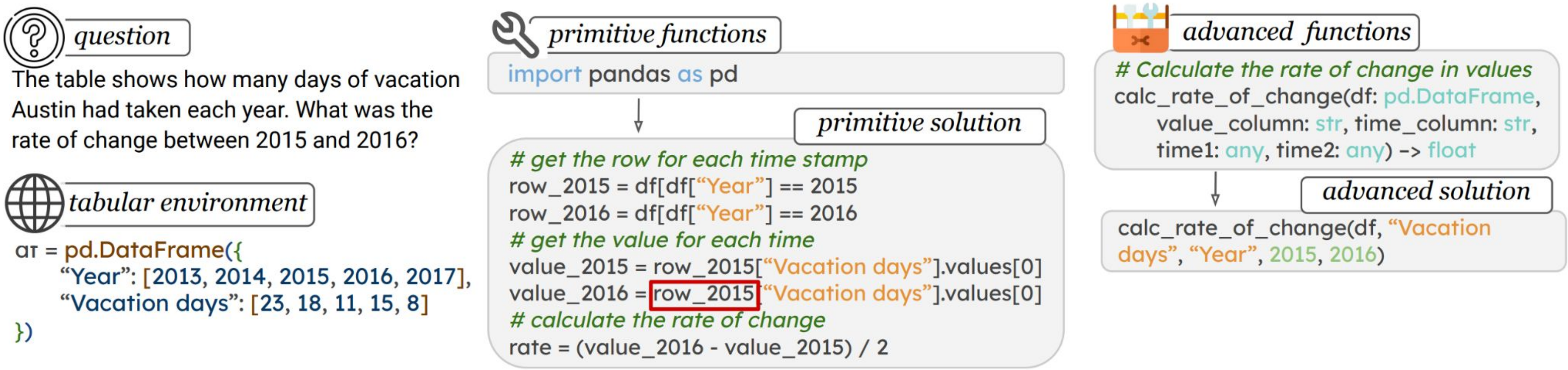
Year	2013	2014	2015
Vacation days	23	18	11

```
import pandas as pd
df = pd.DataFrame({"Year": [2013, 2014, 2015],
"Vacation days": [23, 18, 11]})
max_days = df["Vacation days"].max()
```

Q: Who is wearing the dress?

```
dress_box = locate_objects(image, "dress")
dress_region = crop_region(image, dress_box)
answer = visual_qa(dress_region,
question="Who is wearing the dress?")
```

Why tool making helps?



question

The table shows how many days of vacation Austin had taken each year. What was the rate of change between 2015 and 2016?

tabular environment

```
df = pd.DataFrame({
      "Year": [2013, 2014, 2015, 2016, 2017],
      "Vacation days": [23, 18, 11, 15, 8]
})
```

primitive functions

```
import pandas as pd
```

primitive solution

```
# get the row for each time stamp
row_2015 = df[df["Year"] == 2015
row_2016 = df[df["Year"] == 2016
# get the value for each time
value_2015 = row_2015["Vacation days"].values[0]
value_2016 = row_2015["Vacation days"].values[0]
# calculate the rate of change
rate = (value_2016 - value_2015) / 2
```

advanced functions

```
# Calculate the rate of change in values
calc_rate_of_change(df: pd.DataFrame,
      value_column: str, time_column: str,
      time1: any, time2: any) -> float
```

advanced solution

```
calc_rate_of_change(df, "Vacation
days", "Year", 2015, 2016)
```

Primitive Solution

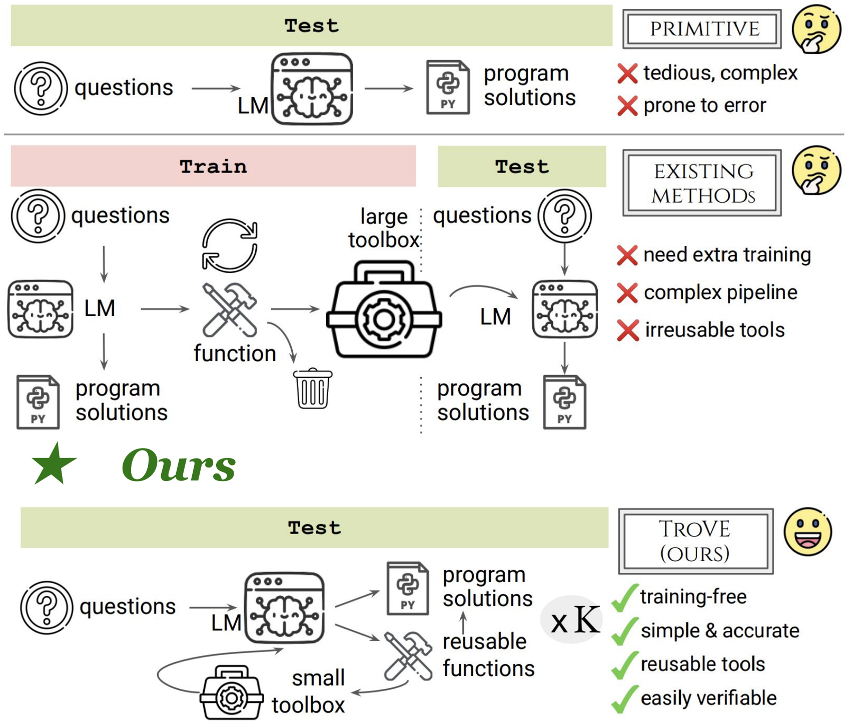
- Tedious, complex
- Error-prone
- Hard to verify

With tools:

- Concise
- Accurate
- Easy to verify

Existing methods are not very efficient...

Adds a ton of computation cost



Test

PRIMITIVE

questions → LM → program solutions

❌ tedious, complex

❌ prone to error

Train

Test

EXISTING METHODs

questions

LM

program solutions

function

large toolbox

questions

LM

program solutions

❌ need extra training

❌ complex pipeline

❌ irreusable tools

★ *Ours*

Test

TROVE (OURS)

questions → LM

program solutions

reusable functions

small toolbox

x K

✓ training-free

✓ simple & accurate

✓ reusable tools

✓ easily verifiable

Tools may not be reusable

question

The table shows how many days of vacation Austin had taken each year. What was the rate of change between 2015 and 2016?

```
def calc_rate(df, time1: int, time2: int):
    # get the row for each time stamp
    row1 = df[df[“Year”] == time1
    row2 = df[df[“Year”] == time2
    # get the value for each time
    value1 = row1[“Vacation days”].values[0]
    value2 = row2[“Vacation days”].values[0]
    # calculate the rate of change
    rate = (value2 - value1) / 1
    return rate
```

new question

The table shows how many words Peter learnt each day What was the rate of change between Jan 1st and Feb 2nd?

How do TroVE make tools?

Pipeline

How do TroVE make tools?

- Using and growing the toolbox
- Agreement-based selection
- Periodic toolbox trimming

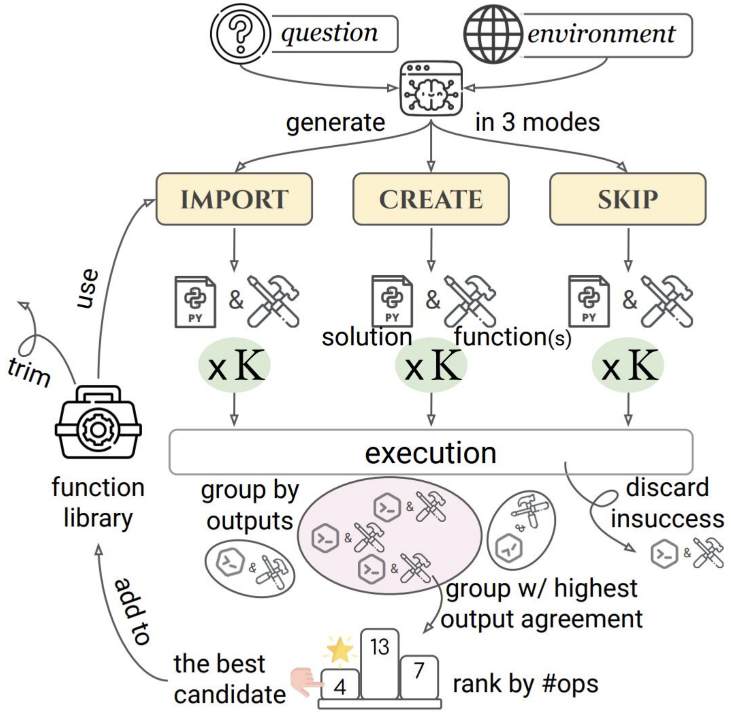
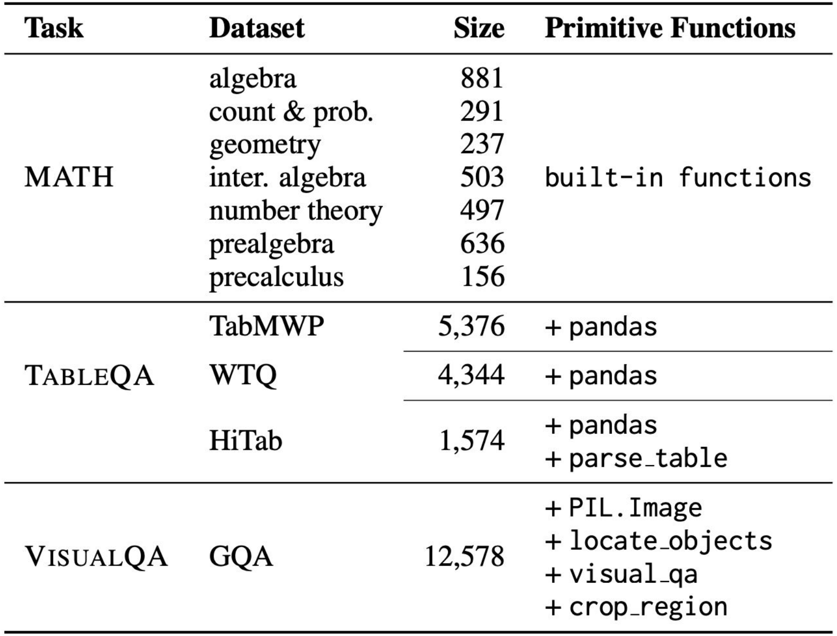


Figure 5. TROVE illustration. Top: generate solutions while using and growing the toolbox. Bottom: select the best response by execution agreement. Left: periodically trim low-utility functions.

Testbed: Dataset & Metrics



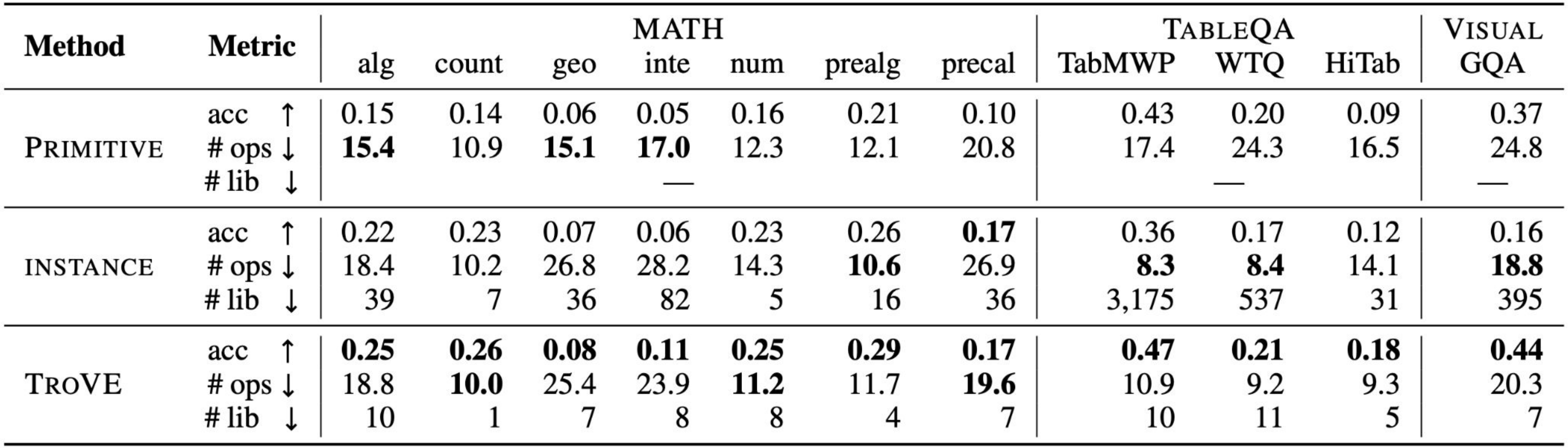
Task	Dataset	Size	Primitive Functions
MATH	algebra	881	built-in functions
	count & prob.	291	
	geometry	237	
	inter. algebra	503	
	number theory	497	
	prealgebra	636	
	precalculus	156	
TableQA	TabMWP	5,376	+ pandas
	WTQ	4,344	+ pandas
	HiTab	1,574	+ pandas + parse_table
VisualQA	GQA	12,578	+ PIL.Image + locate_objects + visual_qa + crop_region

Table 1. Statistics and primitives for three tasks.

Evaluation Metrics

- Answer correctness (acc ↑)
- Solution complexity (#ops ↓)
- Toolbox size (#lib ↓)

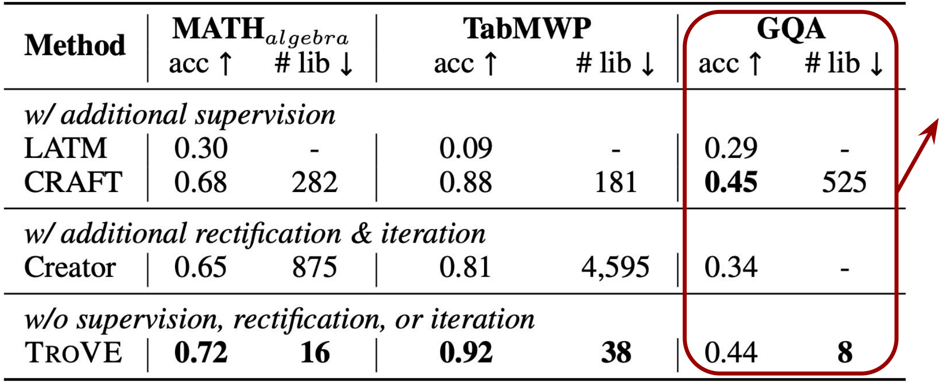
CodeLLaMa: Better Performance with Tools



Method	**Metric**	MATH							TableQA			Visual
		alg	count	geo	inte	num	prealg	precal	TabMWP	WTQ	HiTab	GQA
Primitive	acc ↑	0.15	0.14	0.06	0.05	0.16	0.21	0.10	0.43	0.20	0.09	0.37
	# ops ↓	**15.4**	10.9	**15.1**	**17.0**	12.3	12.1	20.8	17.4	24.3	16.5	24.8
	# lib ↓				—					—		—
Instance	acc ↑	0.22	0.23	0.07	0.06	0.23	0.26	**0.17**	0.36	0.17	0.12	0.16
	# ops ↓	18.4	10.2	26.8	28.2	14.3	**10.6**	26.9	**8.3**	**8.4**	14.1	**18.8**
	# lib ↓	39	7	36	82	5	16	36	3,175	537	31	395
TroVE	acc ↑	**0.25**	**0.26**	**0.08**	**0.11**	**0.25**	**0.29**	**0.17**	**0.47**	**0.21**	**0.18**	**0.44**
	# ops ↓	18.8	**10.0**	25.4	23.9	**11.2**	11.7	**19.6**	10.9	9.2	9.3	20.3
	# lib ↓	10	1	7	8	8	4	7	10	11	5	7

Table 2. CodeLLaMa-7b-Instruct results on MATH, TableQA, and Visual tasks.

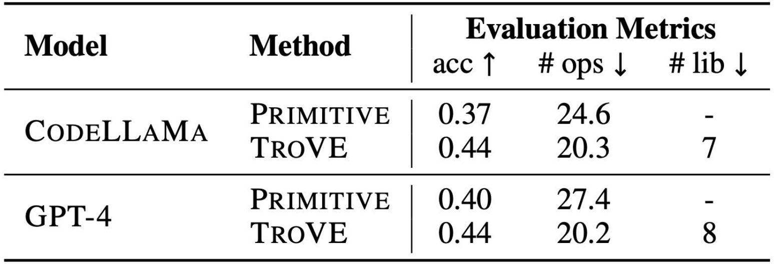
GPT4: better than existing methods



Method	MATH$_{algebra}$ acc ↑	MATH$_{algebra}$ # lib ↓	TabMWP acc ↑	TabMWP # lib ↓	GQA acc ↑	GQA # lib ↓
w/ additional supervision						
LATM	0.30	-	0.09	-	0.29	-
CRAFT	0.68	282	0.88	181	**0.45**	525
w/ additional rectification & iteration						
Creator	0.65	875	0.81	4,595	0.34	-
w/o supervision, rectification, or iteration						
TroVE	**0.72**	**16**	**0.92**	**38**	0.44	**8**

Table 3. Comparing with existing methods using GPT-4. We adopt the baseline results as reported in Yuan et al. (2023). We do not report the *complexity* metric since none of these methods report it (our results in Table 2).

But no better than CodeLLaMa-7B?

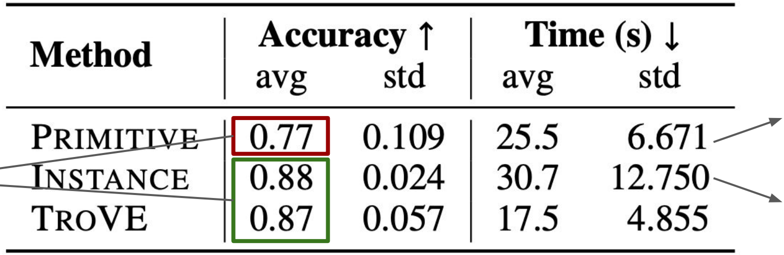


Model	Method	Evaluation Metrics acc ↑	# ops ↓	# lib ↓
CodeLLaMa	Primitive	0.37	24.6	-
	TroVE	0.44	20.3	7
GPT-4	Primitive	0.40	27.4	-
	TroVE	0.44	20.2	8

Table 4. 7B CodeLLaMa2 and GPT-4 perform comparably on the GQA task without training advantage.

Training advantage on primitive functions!

Human Verification: faster, more accurate



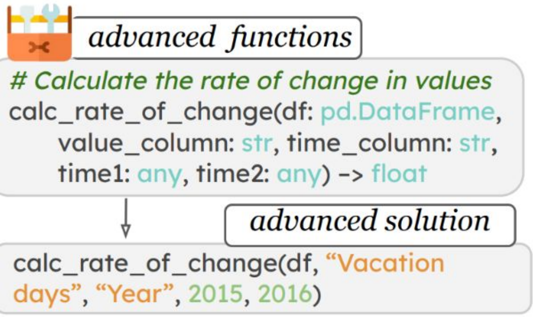
Method	Accuracy ↑		Time (s) ↓	
	avg	std	avg	std
Primitive	0.77	0.109	25.5	6.671
Instance	0.88	0.024	30.7	12.750
TroVE	0.87	0.057	17.5	4.855

Table 5. Human accuracy and time in verifying model-produced solutions with three methods experimented.

10% more accurate

31.4% faster than [Primitive]

43.0% faster than [Instance]

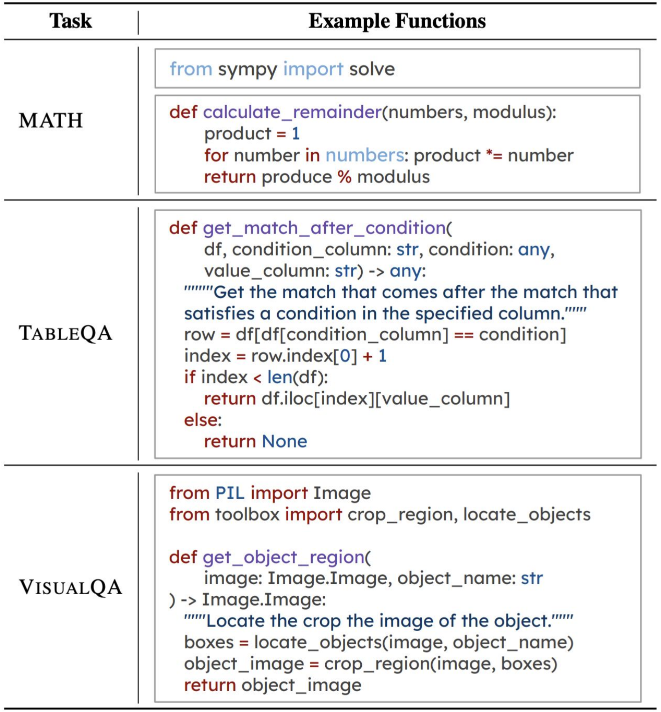


```
# Calculate the rate of change in values
calc_rate_of_change(df: pd.DataFrame,
    value_column: str, time_column: str,
    time1: any, time2: any) -> float
```

```
calc_rate_of_change(df, "Vacation
days", "Year", 2015, 2016)
```

Diverse Tools Across Domains

Varied function types across tasks



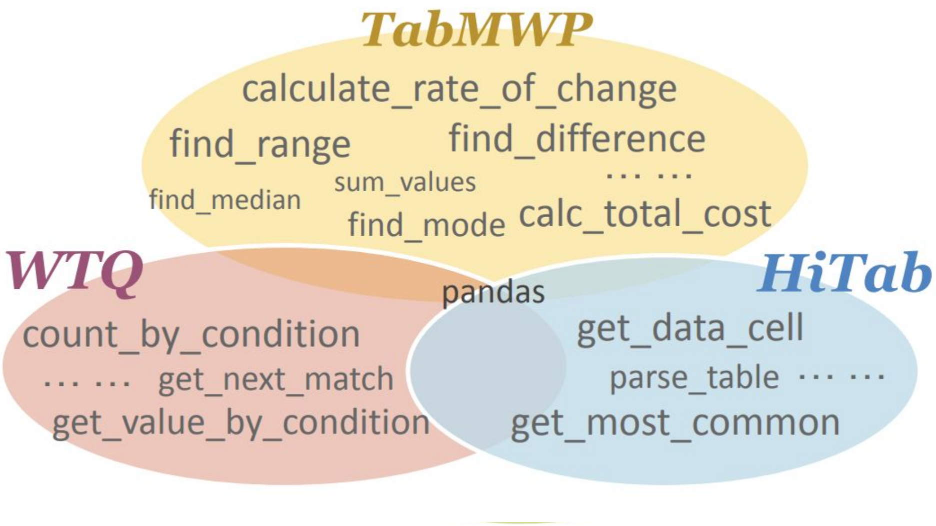
Task	Example Functions

MATH	from sympy import solve
	def calculate_remainder(numbers, modulus):
	product = 1
	for number in numbers: product *= number
	return produce % modulus

TABLEQA	def get_match_after_condition(
	df, condition_column: str, condition: any,
	value_column: str) -> any:
	"""Get the match that comes after the match that satisfies a condition in the specified column."""
	row = df[df[condition_column] == condition]
	index = row.index[0] + 1
	if index < len(df):
	return df.iloc[index][value_column]
	else:
	return None

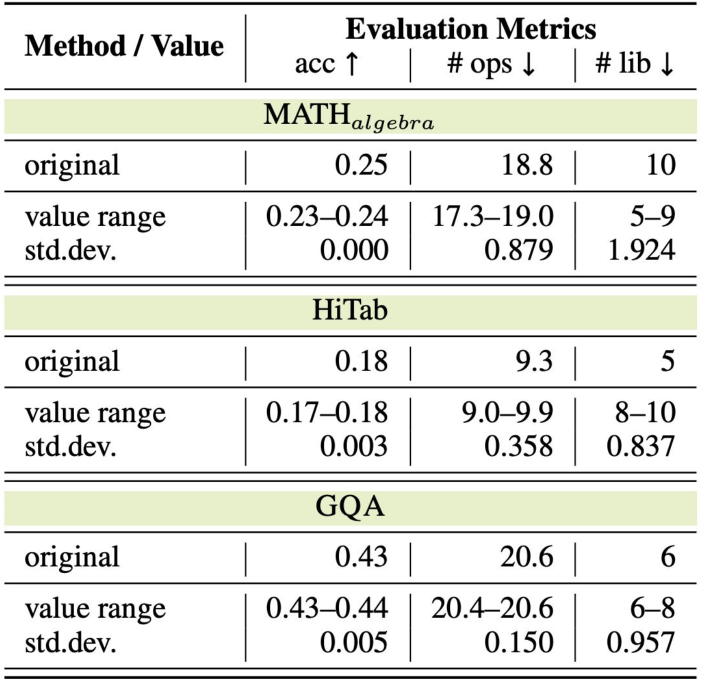
VISUALQA	from PIL import Image
	from toolbox import crop_region, locate_objects
	
	def get_object_region(
	image: Image.Image, object_name: str
	) -> Image.Image:
	"""Locate the crop the image of the object."""
	boxes = locate_objects(image, object_name)
	object_image = crop_region(image, boxes)
	return object_image

Varied functionalities across datasets



Ablation Studies

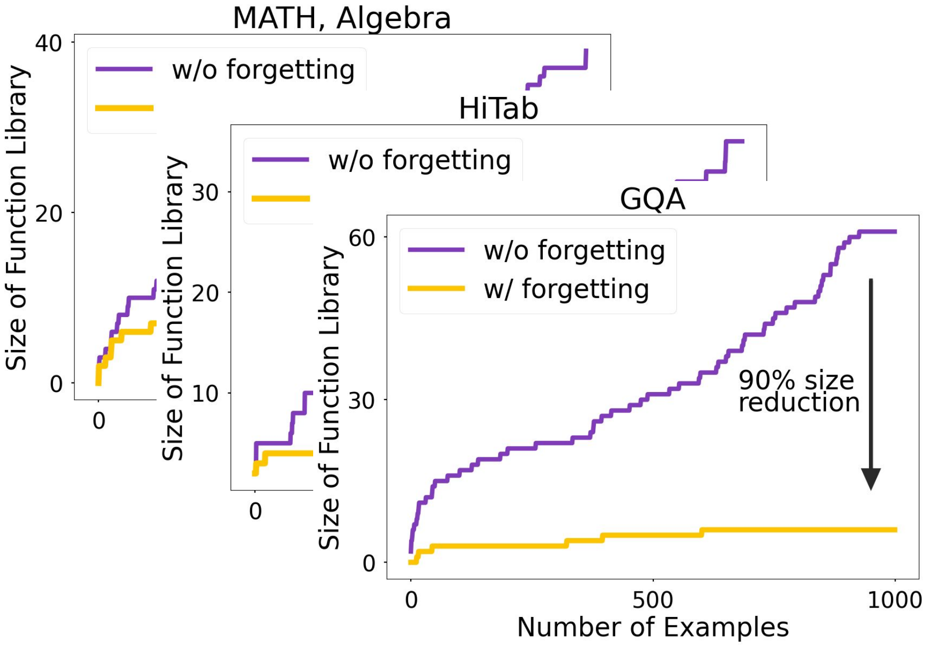
- Robustness to example ordering



Method / Value	Evaluation Metrics acc ↑	 # ops ↓	 # lib ↓
MATH$_{algebra}$			
original	0.25	18.8	10
value range std.dev.	0.23–0.24 0.000	17.3–19.0 0.879	5–9 1.924
HiTab			
original	0.18	9.3	5
value range std.dev.	0.17–0.18 0.003	9.0–9.9 0.358	8–10 0.837
GQA			
original	0.43	20.6	6
value range std.dev.	0.43–0.44 0.005	20.4–20.6 0.150	6–8 0.957

Table 7. CODELLAMA results with alternative orders.

- Necessity of periodic toolbox trimming



Recap: TroVE

- Make tools for programmatic tasks
- Get more accurate, concise solutions
- Facilitates human verification
- Naturally adaptive to various tasks/domains

Thank You!