Proposed Projects: Common theme

Building an Adaptive and Modular Memory To Solve (Knowledge-based) Reasoning Tasks

SWP 2025/26: Anette Frank

Test-Time Inference with Adaptive Memory To Solve (Knowledge-based) Reasoning Tasks

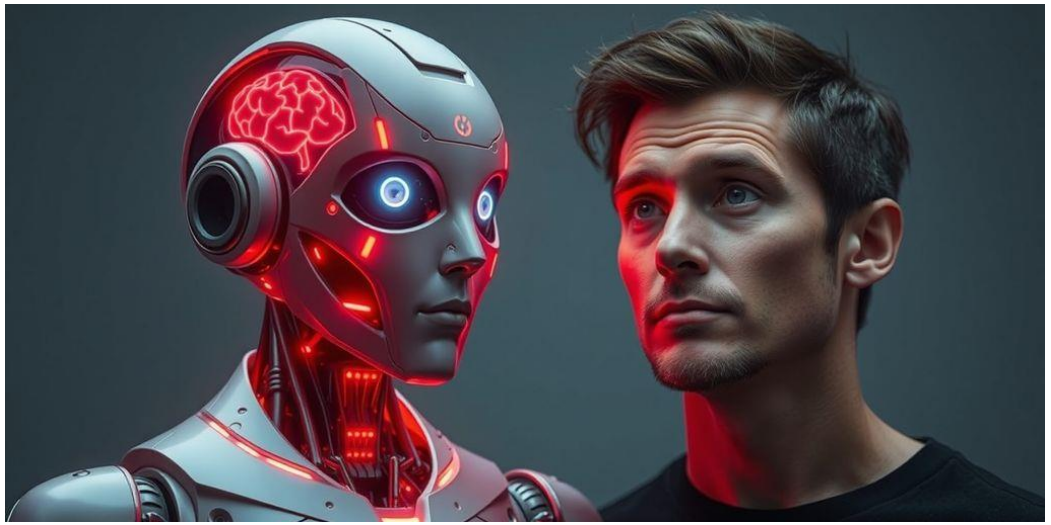
LLMs process enormous amounts of information during pre-training, but:

- can they ***generalize*** from it?
- can they ***preserve*** acquired knowledge?
- can they ***retrieve*** it on demand?

➥ Memorization and generalization ability?

- LLMs suffer from catastrophic forgetting (Fu & Frank 2024)
- Learning requires challenging tasks but

  Test-time inference does not profit from 'experience'



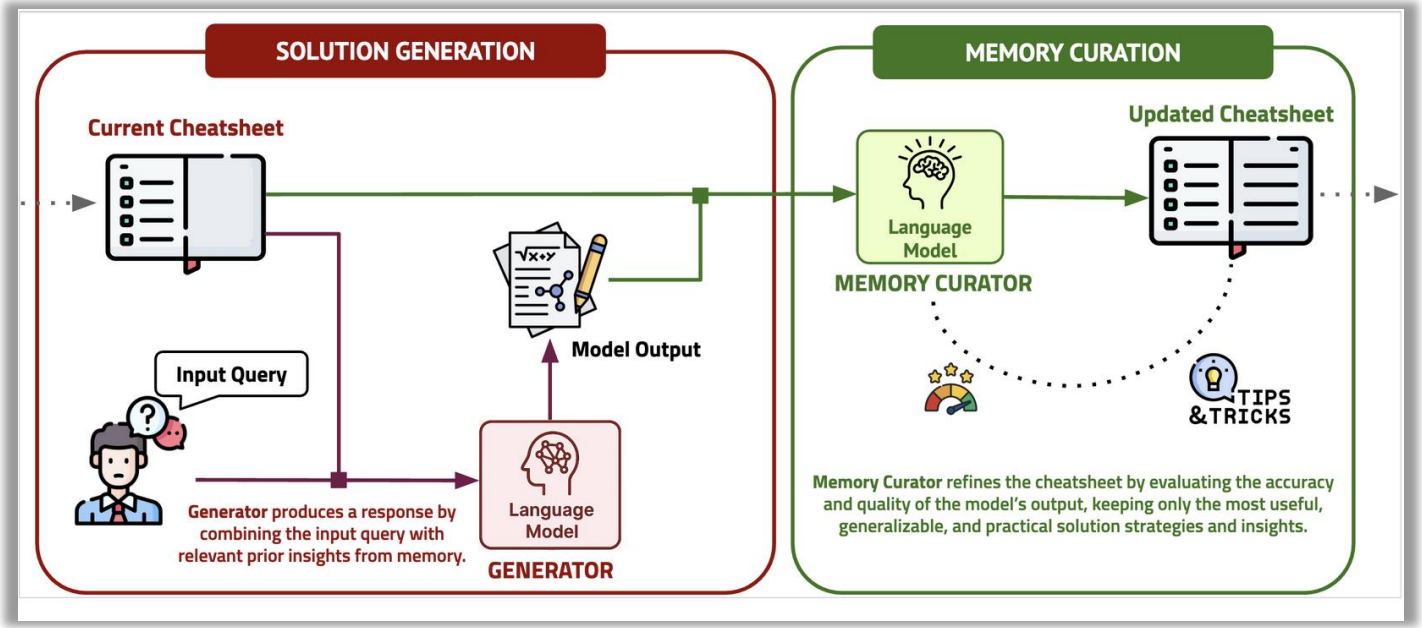
When Machines Remember Better Than Humans: The AI Memory Advantage (but we are not yet there ...)

How to provide an **adaptive memory** to LLMs for **efficient test-time inference**?

- Many approaches are being explored (Wu et al. 2025)

Dynamic Cheatsheet – Test-Time Learning with Adaptive Memory

- Equip black-box LLMs w/
  - **persistent**,
  - **evolving memory**
  - at **inference** time



➥ augment LLM experience

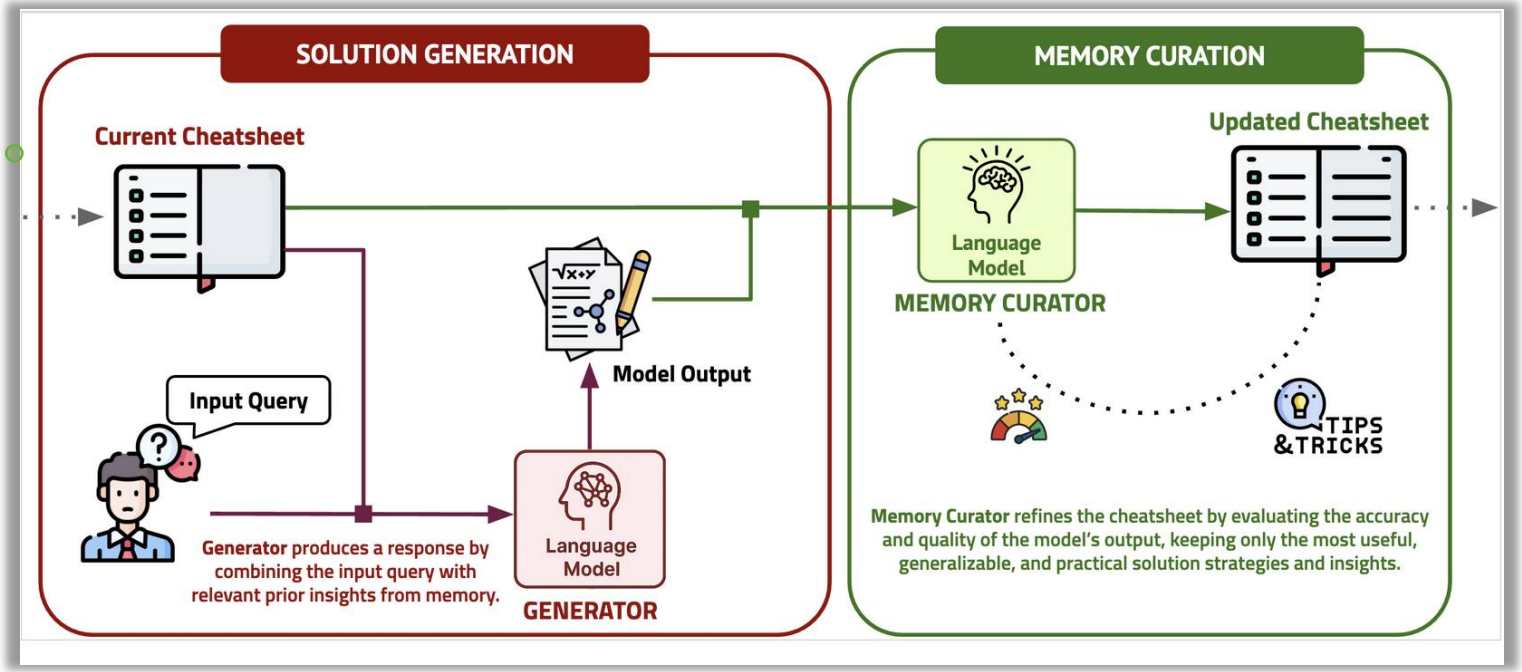
➥ no model updates;
instead: extending and refining memory

➥ preserve refined memory across sessions

Suzgun, M., Yuksekgonul, M., Bianchi F., Jurafsky, D., Zo, J. (2025):
Dynamic Cheatsheet: Test-Time Learning with Adaptive Memory, arXiv

Key Features

Augment, refine (curate) and *use* an evolving cheatsheet during test-time



- **Persistent Memory**: grow & use **knowledge base (memory)** during inference
- **Self-Curated Storage**: concise, transferable information vs. full transcripts
- **Black-Box Compatible**: requires no access to model parameters
- **Zero-Shot Learning**: improves w/o supervision or human feedback
- **Experience-Driven Learning**: isolated inference events ✗ ⟺ cumulative learning ✓

Dynamic Cheat Sheet (DC) Variants

1. DC-Cumulative (DC-Cu):

A **growing cheat-sheet accumulates knowledge** from all queries.

- Good for sequential problem solving
- Insights can build upon each other (➥ curriculum learning)

2. DC-Retrieval & Synthesis (DC-RS):

a. Similarity-based **retrieval**: find prior experiences

b. **Synthesize them** into a cheatsheet for each query.

- Ideal for tasks with diverse queries

Dynamic Cheatsheet (DS) – Variants and Steps

x_i

BL (Baseline)

1: **for** $i \in [1, \ldots, n]$ **do**
2: $\quad \tilde{y}_i = \text{Gen}(x_i)$ ▷ *Solution generation*
3: **end for**

y_i

Baseline QA or Conversational Setup:
for each *input* x_i , the model generates an *answer* y_i in test time inference

Dynamic Cheatsheet (DS) – DC-Ø$_{\text{(empty memory)}}$

x_i

DC-∅ (Empty Memory)

1: **for** $i \in [1, \ldots, n]$ **do**
2: $M_i = \emptyset$ ▷ *Empty memory*
3: $\tilde{y}_i = \text{Gen}(x_i, M_i)$ ▷ *Solution generation*
4: **end for**

y_i

Memory (empty, not used): similar to BL, since M = Ø

for each *input* x_i , the model generates an *answer* y_i in test time inference

Dynamic Cheatsheet (DS) – DC-Cu$_{\text{Curation}}$

Cur does not have access to ground truth, but ..

x_i

DC-Cu. (Cumulative)

1: $M_0 \leftarrow \emptyset$ ▷ *Memory initialization*
2: **for** $i \in [1, \ldots, n]$ **do**
3: $\tilde{y}_i = \mathsf{Gen}(x_i, M_{i-1})$ ▷ *Solution generation*
4: $M_i = \mathsf{Cur}(M_{i-1}, x_i, \tilde{y}_i)$ ▷ *Memory curation*
5: **end for**

y_i

Memory Curation **Cur (M_{i-1}, x_i, y_i)** using **M_{i-1} from previous interactions**
for each *input* x_i, the model generates *answer* y_i in test time inference,

- using knowledge M_{i-1} gained from prior experience, and
- integrating x_i, y_i with M_{i-1} to yield new memory state M_i

… can ***use tools and models*** to ***verify*** the validity of the solution and ***transform it*** into ***generalizable and efficient strategies*** or code.

Cur (i) **distills** *useful and generalizable answers* into a form suitable for later reference,
(ii) ***refines or removes*** *existing memory* entries (if they are superseded by new solutions,
(iii) ***consolidates*** *memory entries* to *retain* ***succinct, high-impact*** *references and heuristics*.

Dynamic Cheatsheet (DS) – DC-Cu$_{\text{Curation}}$

x_i

DC-Cu. (Cumulative)

$M_0 \leftarrow \emptyset$ ▷ *Memory initialization*

for $i \in [1, \ldots, n]$ **do**

  $\tilde{y}_i = \mathsf{Gen}(x_i, M_{i-1})$ ▷ *Solution generation*

  $M_i = \mathsf{Cur}(M_{i-1}, x_i, \tilde{y}_i)$ ▷ *Memory curation*

end for

y_i

Weaknesses of DC-Cc:

1. memory is updated ***after*** query processing; is ***not*** refined ***before*** response generation
2. does not store or revisit ***past input-output pairs*** unless explicitly retained in memory

Dynamic Cheatsheet (DS) – DC-RS$_{\text{Retrieval\&Synthesis}}$

x_i

DR (Dynamic Retrieval)

1: **for** $i \in [1, \ldots, n]$ **do**
2: $R_i = \texttt{Retr}(x_i, \{(x_j, \tilde{y}_j)\}_{j<i}, k)$ ▷ *Retrieval*
3: $M_i = R_i$ ▷ *Memory contains only select examples*
4: $\tilde{y}_i = \texttt{Gen}(x_i, M_i)$ ▷ *Solution generation*
5: **end for**

y_i

DC-RS: Retrieval (Retr) & Synthesis

(1) retrieves *top-k (k=3) most similar inputs,* with model-generated outputs from seen samples to yield **R_i(k)** (or **R_i**).

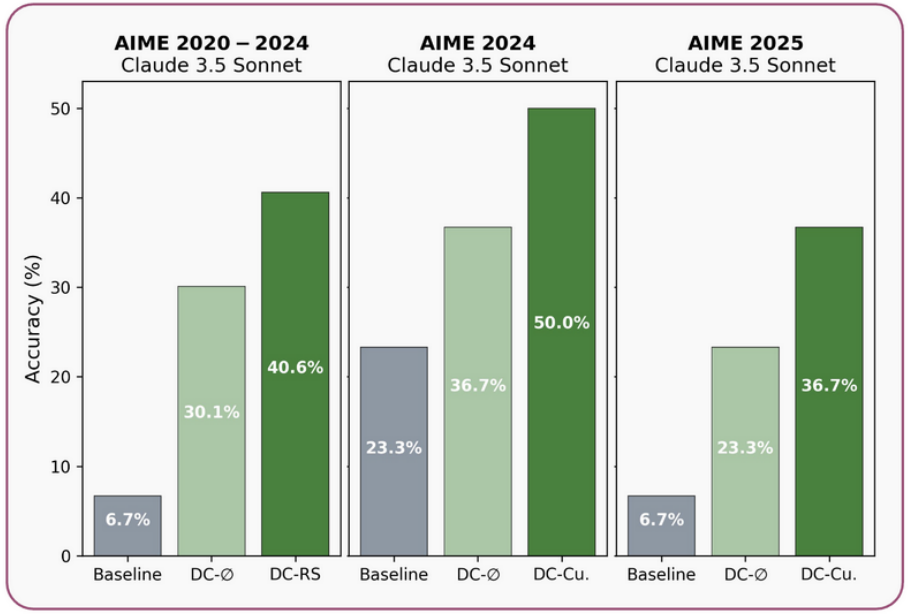
(2) jointly with M_{i-1}, R_i is passed to curator **Cu** to update the memory, yielding M_i.

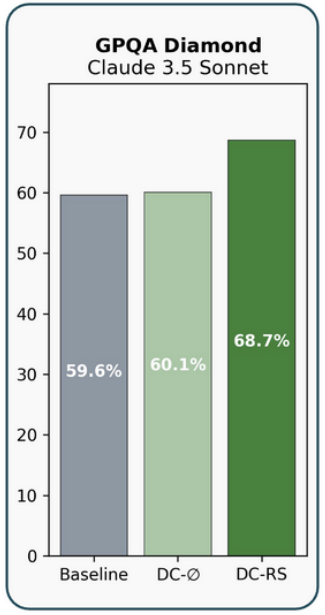
(3) generator uses M_i to produce y_i, given x_i and M_i.

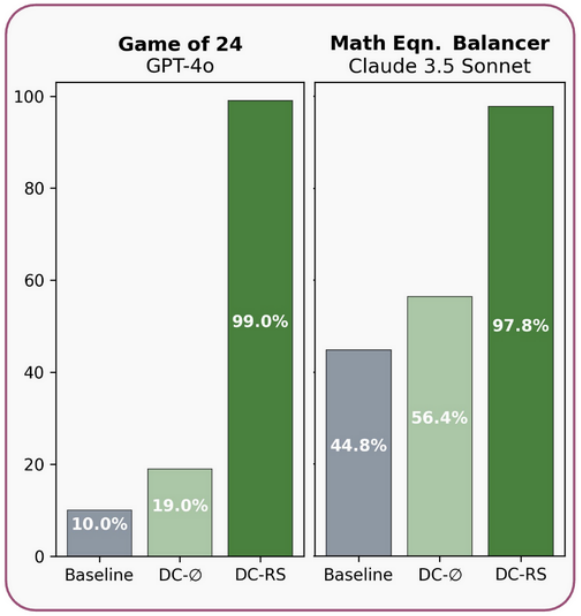
$$R_i = \texttt{Retr}(x_i, \{(x_j, \tilde{y}_j)\}_{j<i}, k)$$
$$M_i = \texttt{Cur}(M_{i-1}, x_i, R_i)$$
$$\tilde{y}_i = \texttt{Gen}(x_i, M_i)$$

Performance Improvements

- **Mathematics**: Claude 3.5 Sonnet: **retains algebraic insights** on AIME math exams
- **Puzzles**: GPT-4o: Discovers and reuses Python-based solutions on Game of 24
- **Arithmetic**: Near-perfect accuracy on tasks like balancing equations
- **Knowledge-Intensive Tasks**: GPQA-Diamond; MMLU-Pro Engineering & Physics problems







Examples

Reusable Code Snippets and Solution Strategies
<memory_item>

<description>
Game 24 Solver Strategy: Solve the 24 Game by systematically testing combinations of four numbe
arithmetic operations (+, -, *, /) and parentheses to achieve a result of 24. Each number must be use
</description>

<example>
Steps:
1. Understand the Problem:
- Input: Four integers.
- Goal: Combine the numbers using arithmetic operations and parentheses to evaluate to 24.
- Constraints: Each number must be used exactly once.
2. Approach:
- Use brute force or systematic trial-and-error to test all possible combinations of numbers, operati
parentheses.
- Prioritize operations that simplify the problem (e.g., division resulting in integers, subtraction rec
- Check edge cases (e.g., repeated numbers, large/small values).
3. Example Solution:
- Input: 6, 8, 8, 12
- Solution: (6 * ((8 + 8) - 12) = 24)
4. Tips:
- Use parentheses to control operation precedence.
- Division should result in valid integers or fractions.
- Test all permutations of the numbers and operations systematically.
Python Code for Automation

```
from itertools import permutations, product
```

Python Code for Automation

```
from itertools import permutations, product

def solve_24(numbers):
    ops = ['+', '-', '*', '/']
    for nums in permutations(numbers):
        for opers in product(ops, repeat=3):
            expressions = [
                f"({nums[0]} {opers[0]} {nums[1]}) {opers[1]} ({nums[2]} {opers[2]} {nums[3]})",
                f"(({nums[0]} {opers[0]} {nums[1]}) {opers[1]} {nums[2]}) {opers[2]} {nums[3]}",
                f"({nums[0]} {opers[0]} ({nums[1]} {opers[1]} {nums[2]})) {opers[2]} {nums[3]}",
                f"{nums[0]} {opers[0]} (({nums[1]} {opers[1]} {nums[2]}) {opers[2]} {nums[3]})",
                f"{nums[0]} {opers[0]} ({nums[1]} {opers[1]} ({nums[2]} {opers[2]} {nums[3]}))"
            ]
            for expr in expressions:
                try:
                    if abs(eval(expr) - 24) < 1e-6:  # Check if result is 24
                        return expr
                except ZeroDivisionError:
                    continue
    return "No solution"

# Example Usage
print(solve_24([6, 8, 8, 12]))  # Output: 6 * ((8 + 8) - 12)
```

</example>

</memory_item>
Count: 99

Figure 5: Excerpt from GPT-4o's external memory after processing 100 examples from Game of 24 under DC-RS. Early in the test sequence, the model discovered a Python-based brute-force solution, stored it, and subsequently retrieved it for subsequent puzzles. This shift to structured code reuse resulted in a dramatic performance increase from 10% to 99% accuracy, eliminating arithmetic errors and redundant problem-solving efforts.

Examples

GENERAL META-REASONING STRATEGIES

<memory_item>

<description>

Systematic Problem Analysis Framework (*Reference: Q1-Q20*)

For complex mathematical problems:

1. State problem requirements clearly
2. List key observations and theorems applicable
3. Identify patterns and relationships
4. Break into manageable sub-problems
5. Verify against examples
6. Consider computational approach when analytical solution is complex
7. For grid problems, analyze movement patterns and symmetries
8. For combinatorial problems, use appropriate counting techniques
9. Implement verification code when possible
10. Consider edge cases and constraints
11. For grid coloring problems, consider row/column patterns

</description>

<example>

Example application:

1. Requirements: list all given conditions
2. Observations: identify applicable theorems
3. Patterns: look for structural relationships
4. Sub-problems: break into steps
5. Verification: test against examples
6. Implementation: use Python for verification

</example>

</memory_item>

Count: 20

Figure 6: Example of Claude 3.5 Sonnet's curated memory after processing 20 AIME 2024 questions under DC-Cu. The memory captures key solution strategies, enables the model to generalize across similar computational problems, and boosts its accuracy.

Performance Improvements

Dynamic Cheatsheet: Test-Time Learning with Adaptive Memory

Tasks	Claude 3.5 Sonnet					GPT-4o				
	BL	DC-∅	DR	DC-Cu.	DC-RS	BL	DC-∅	DR	DC-Cu.	DC-RS
AIME 2024	23.3	36.7	43.3	**50.0**	46.7	20.0	36.7	26.7	36.7	**40.0**
AIME 2025	6.7	23.3	23.3	**36.7**	30.0	6.7	10.0	10.0	16.7	**20.0**
AIME 2020–24	6.7	30.1	39.1	38.4	**40.6**	9.8	24.1	24.1	20.3	**24.8**
Game of 24	12.0	10.0	11.0	**14.0**	**14.0**	10.0	19.0	6.0	93.0	**99.0**
GPQA Diamond	59.6	60.1	63.6	61.1	**68.7**	**57.1**	57.1	55.1	58.1	57.1
Math Eqn. Balancer	44.8	56.4	60.4	**100**	97.8	50.0	88.0	**100**	**100**	99.2
MMLU Pro Eng.	61.2	57.2	65.2	66.8	**67.6**	**53.2**	51.6	48.8	44.0	51.2
MMLU Pro Physics	74.0	75.6	80.4	77.6	**82.0**	75.6	70.8	**75.6**	70.4	75.2

Table 1: Performance comparison of Dynamic Cheatsheet (DC) variants for Claude 3.5 Sonnet and GPT-4o across multiple benchmarks. **BL** (Baseline): standard inference without memory; **DC-∅** (Empty Memory): includes structured problem-solving and explicit tool-use instructions but no memory retention mechanism; **DR** (Dynamic Retrieval): uses retrieval but lacks curated memory updates; **DC-Cu** (Cumulative Memory): iteratively accumulates model solutions but lacks retrieval; and **DC-RS** (Retrieval & Synthesis): combines retrieval with memory refinement/synthesis. These results highlight substantial accuracy gains under DC: Claude 3.5 Sonnet's AIME 2024 accuracy jumps by 27% under DC-Cu, and GPT-4o's Game of 24 accuracy leaps from 10% to 99% under DC-RS.

GPQA: A Graduate-Level Google-Proof Q&A Benchmark

Chemistry (general)

A reaction of a liquid organic compound, which molecules consist of carbon and hydrogen atoms, is performed at 80 centigrade and 20 bar for 24 hours. In the proton nuclear magnetic resonance spectrum, the signals with the highest chemical shift of the reactant are replaced by a signal of the product that is observed about three to four units downfield. Compounds from which position in the periodic system of the elements, which are also used in the corresponding large-scale industrial process, have been mostly likely initially added in small amounts?

A) A metal compound from the fifth period.
B) A metal compound from the fifth period and a non-metal compound from the third period.
C) A metal compound from the fourth period.
D) A metal compound from the fourth period and a non-metal compound from the second period.

Organic Chemistry

Methylcyclopentadiene (which exists as a fluxional mixture of isomers) was allowed to react with methyl isoamyl ketone and a catalytic amount of pyrrolidine. A bright yellow, cross-conjugated polyalkenyl hydrocarbon product formed (as a mixture of isomers), with water as a side product. These products are derivatives of fulvene. This product was then allowed to react with ethyl acrylate in a 1:1 ratio. Upon completion of the reaction, the bright yellow color had disappeared. How many chemically distinct isomers make up the final product (not counting stereoisomers)?

A) 2
B) 16
C) 8
D) 4

number of chromosomes,

cies.

which contributes to heat
ate a heat-tolerant cultivar
ng its accumulation under
tivar they study. There are

of translation and before

processing in ER
B) Trimethylation of lysine of H3 histone in position 27 at the promoter of the gene encoding the target protein
C) A stop-codon occurs in the 5'-UTR region of the gene encoding the target protein
D) The proteolysis process disrupts a quaternary structure of the protein, preserving only a tertiary structure

GPQA: A Graduate-Level Google-Proof Q&A Benchmark

Genetics

If a sperm from species A is injected into an egg from species B and both species have the same number of chromosomes, what would be the main cause of the resulting zygote mortality?
A) Species specific zona pellucida proteins on the egg cannot bind sperms from a different species.
B) Epistatic interactions between the genes of different species
C) Chromosomal incompatibilities will cause failure of meiosis leading to death of zygote.
D) Chromosomal recombination will not occur in different species.

Molecular Biology

A scientist studies the stress response of barley to increased temperatures and finds a protein which contributes to heat tolerance through the stabilisation of cell membrane. The scientist is very happy and wants to create a heat-tolerant cultivar of diploid wheat. Using databases, they find a heat tolerance protein homologue and start analysing its accumulation under heat stress. Soon enough, the scientist discovers this protein is not synthesised in the wheat cultivar they study. There are many possible reasons for such behaviour, including:
A) A miRNA targets the protein, which makes exonucleases cut it immediately after the end of translation and before processing in ER
B) Trimethylation of lysine of H3 histone in position 27 at the promoter of the gene encoding the target protein
C) A stop-codon occurs in the 5'-UTR region of the gene encoding the target protein
D) The proteolysis process disrupts a quaternary structure of the protein, preserving only a tertiary structure

Observed problems and possible solutions (Suzgun et al. 2025)

Long-context generation versus understanding.

- Memory curation can demand ***precise reproduction or modification of prior knowledge***.
- Models sometimes ***only references or abbreviates existing memory*** instead of rewriting it.
  Such truncated memory updates can reduce the quality of stored heuristics over time.
- **Solution**: maintain a ***structured, external KB*** that the LM can reference w/o regenerating text each time

Retrieval bottlenecks and noise

- **Poorly filtered retrieval** mechanisms can introduce confusion, especially with diverse or loosely related queries.
- GPT-4o's performance occasionally dipped in GPQA-Diamond due to **suboptimal retrieval choices**.
- Need robust retrieval methods that select higher-quality exemplars.

Hierarchical and modular memory

- Specialized domains may benefit from **subdividing or hierarchically organizing memory**.
- System could maintain separate curated **memories for specific topics/domains** with specialized retrieval or curation mechanisms. This could reduce the load on a unified memory and help isolate errors within domains.

Insights from DC, and how to improve in future work

‼ Model capabilities

‼ Structured memories ➥ **Project I**

‼ Modularization ➥ **Project II**

Your projects can take the next steps:

➥ **Structured Memory**: KG construction, augmentation and refinements

- Improve retrieval and generalization

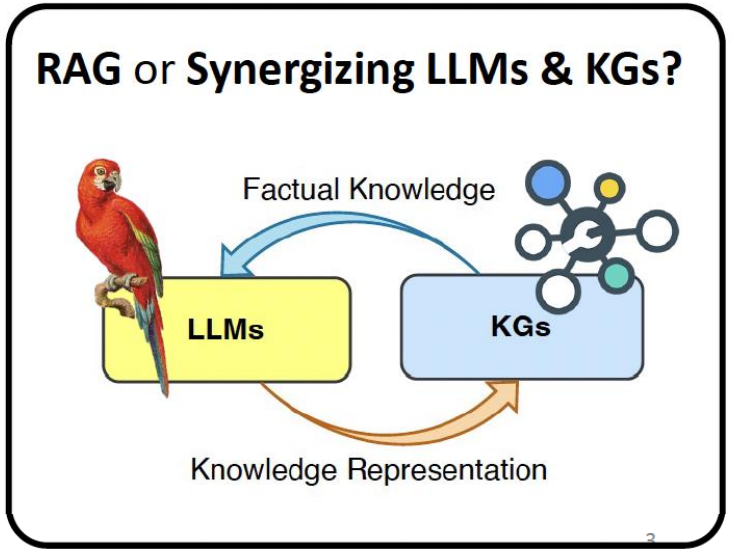
➥ **Modularizing Memory**: Modularization using Mixture-of-Expert models

- Improve interpretability, domain specificity and performance

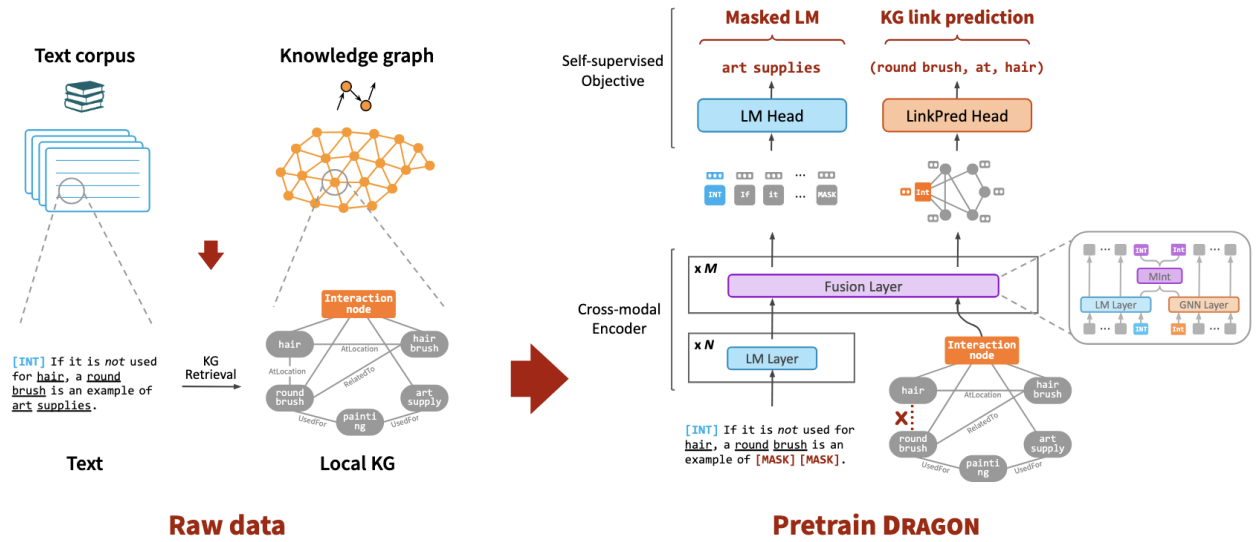
Project 1: Adaptive Structured Memory

How to?

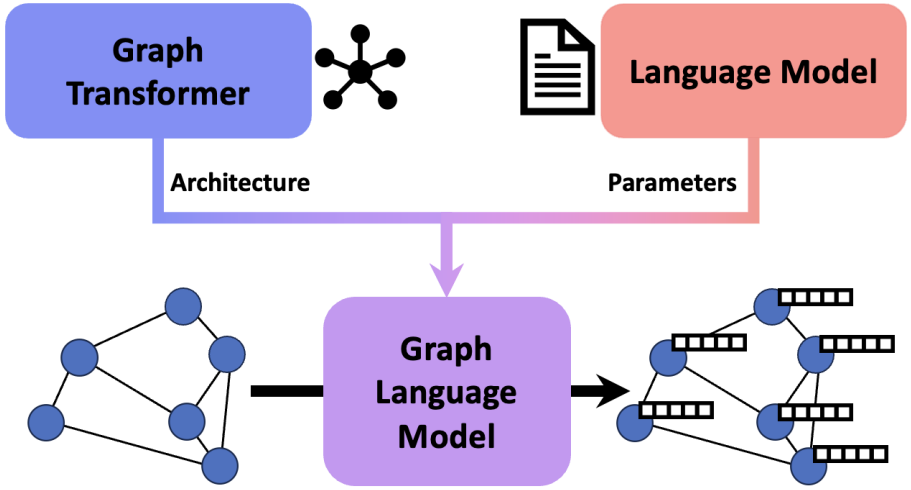
- Text & KG in pre-training (+ FT on adapted memory?) (Dragon)
- Joint inference$_{\text{Text \& KB}}$ in training & inference
- Graph-RAG (e.g., Plenz et al. 2023: CCKG)
- Graph Language Model (Plenz & Frank 2024)



Unifying LLMs and KGs: A Roadmap 2024
access via Heidelberg University



Dragon: pre-training only



GLM: pre-trained LLM + GLM FT on text & subKG

An Adaptive Structured Memory for Memory-augmented Reasoning Tasks

- **Method**
  - Select task datasets & resources & produce **model explanation**s (ICL or Reasoning models)
  - Use data and model explanations **to construct a KG** from dataset samples (train)
  - Design **curation and retrieval methods** (graph augmentation, refinements, resolving inconsistencies)
- **Datasets & Tasks**
  - ToM, SocialIQA, α-NLI/NLG, Commonsense QA
  - bAbI, dyna-bAbI tasks
  - Frodo / Refiner ; ExplaGraphs
- **Evaluation**
  - **Ablations:** DC +/- adaptive +/- structured Memory
  - Model **performances & qualitative analysis**
  - Graph Construction evaluation: MINE; Graph Metrics
    - Knowledge resources for intrinsic evaluation (see Proj II)

- Hypothesis-Driven Theory-of-Mind Reasoning for Large Language Models
- Dyna-bAbI: unlocking bAbI's potential with dynamic synthetic benchmarking
- Making Reasoning Matter: Measuring and Improving Faithfulness of Chain-of-Thought Reasoning
- ExplaGraphs: An Explanation Graph Generation Task for Structured Commonsense Reasoning
- ... See also Proj II

Knowledge Graph Construction Methods & Tools

Text2KG (Lairgi et al., 2024)

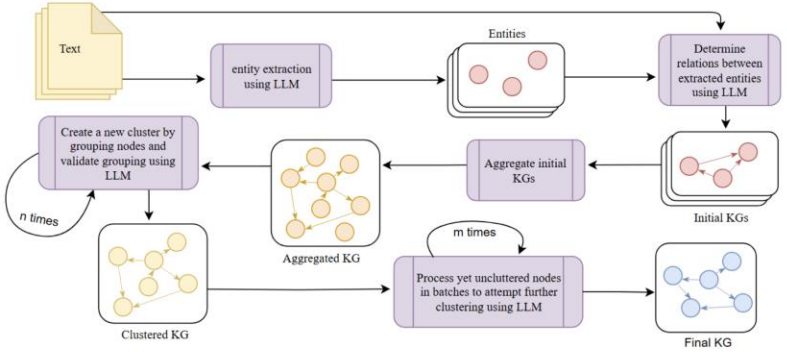
- Incremental, topic-independent zero-shot KG construction using LLMs
- Modules:
  - Document Distiller;
  - Incremental Entity Extractor;
  - Incremental Relation Extractor;
  - Graph Integrator and Visualization.

SAC-KG (Chen et al., 2024)

- Construct multi-level KGs using LLMs
- Modules: Generator, Verifier, and Pruner.
  - **Generator**: extract single-level KG by producing relations and tails from text
  - **Verifier & Pruner**: jointly correct generation errors & **trigger iterations** for **next-level KG**.

KGGen (Mo et al., 2025)

- KGGen: cluster entities to reduce sparsity [available as Python library]
  - **Generate:** first *entity*, then *relation extraction*
  - **Aggregate**: Combine unique entities & edges in a single KG.
  - **Cluster** i) entities, then ii) edge using LLMs

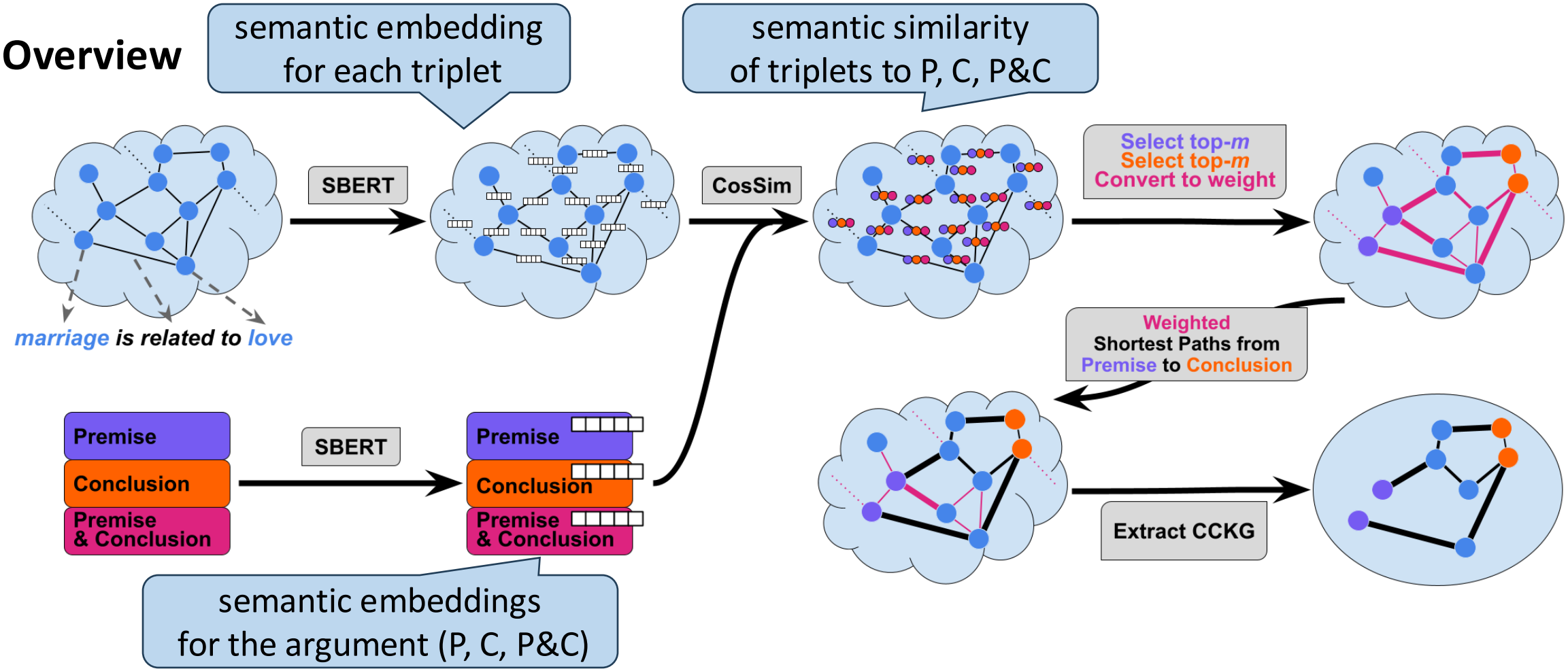


- MINE:
  - Benchmark for KG construction quality
  - KGGen shows superior performance

Retrieval from existing KGs

Similarity-weighted Construction of Contextualized CS KGs (CCKG)

Overview

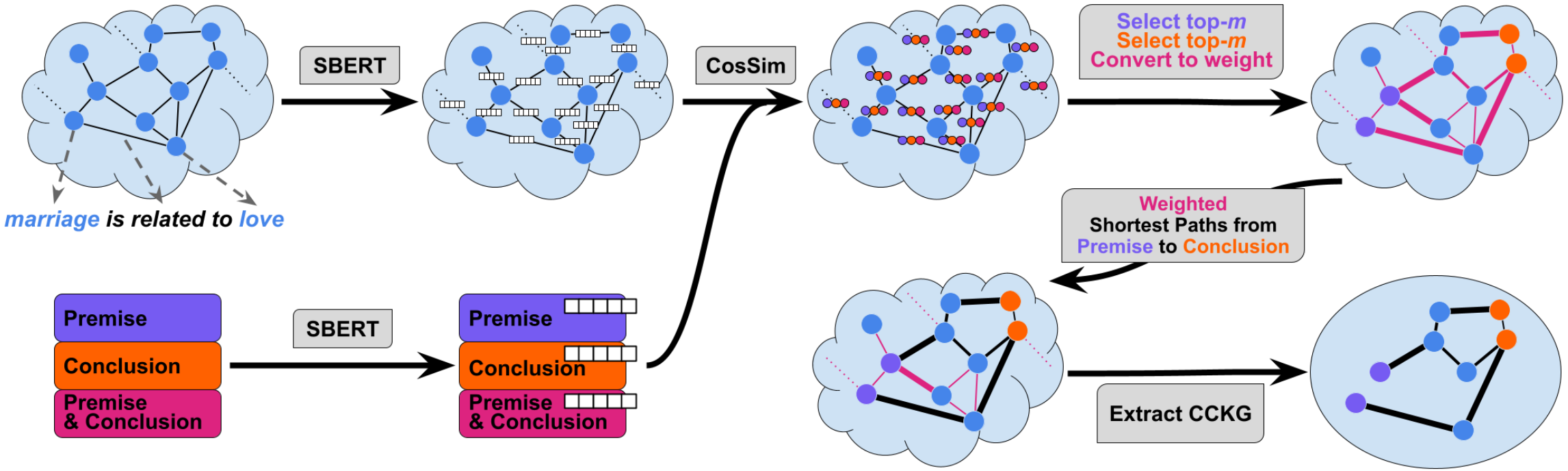


Plenz et al. 2023: Similarity-weighted Construction of Contextualized Commonsense Knowledge Graphs for Knowledge-intense Argumentation Tasks, ACL.

CCKG Construction Method

select concepts from most similar triplets; determine edge weights

Overview



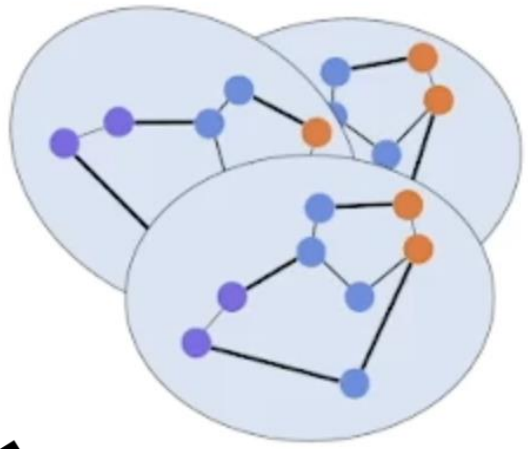
Get shortest paths from P to C that maximize similarity between path and argument;
path includes similar (relevant) intermediate concepts

Evaluation

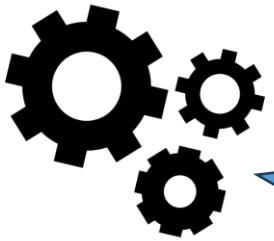
task agnostic

ExplaGraph: Predicting Explanatory Graph Structure from Argument premise & Conclusion

Intrinsic



High precision & recall of implicit CSK

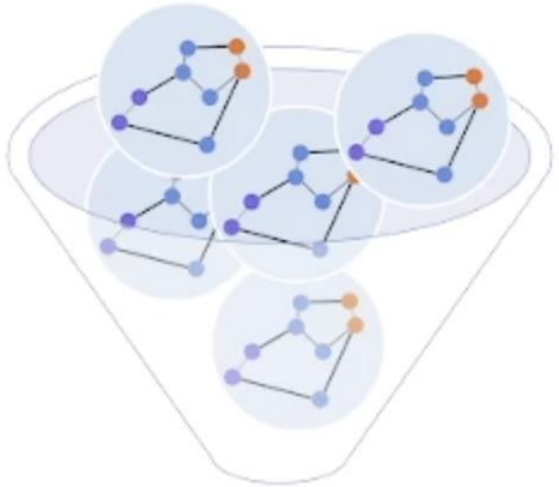


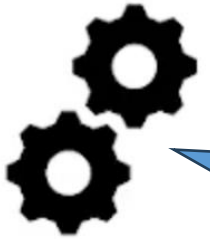
Outperforming supervised SOTA

Beneficial in downstream tasks

Extrinsic

ValNov: Predicting Validity and Novelty of Arguments



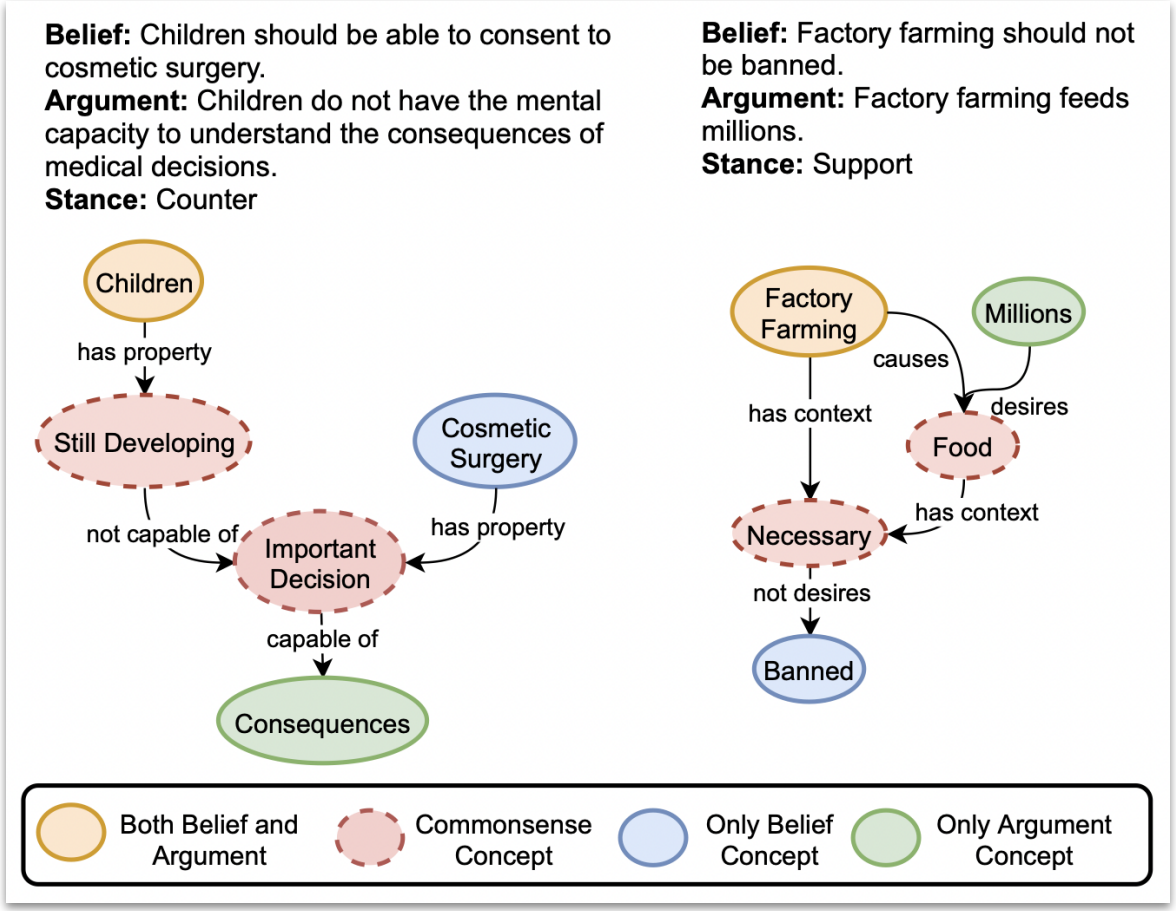


Comparable to GPT3-based system

ExplaGraphs — Saha *et al.* 2021

Description

- **Task**: **Predict Graphs** and Stance
  - ***Argument*** can be seen as ***Premise***
  - ***Belief*** can be seen as ***Conclusion***
- **Goldgraphs**
  - Relations similar to CN
  - Concepts are different from CN
  - We create **Artificial KG *ExplaKnow*** by combining all train- and dev-goldgraphs



Saha *et al.* 2021, Figure 1

Saha et al. 2021: [ExplaGraphs: An Explanation Graph Generation Task for Structured Commonsense Reasoning](). EMNLP

ExplaGraphs

ExplaKnow — Intrinsic Evaluation

Configuration		#Concepts	Concept $F1 \uparrow$	Triplet $F1 \uparrow$	GED $\downarrow$	G-BS $\uparrow$
CCKG	$m = 1$	3.9	**41.7**	**21.0**	**0.35**	64.8
	$m = 2$	6.7	37.9	19.9	0.39	**72.3**
	$m = 3$	9.4	34.4	18.1	0.44	63.8
	$m = 4$	12.2	31.4	16.6	0.50	53.6
	$m = 5$	14.9	28.8	15.1	0.55	46.0
Superv.	T5	4.5	**47.2**	**3.8**	**0.33**	**76.3**
	RE-SP	5.9	42.9	1.2	0.37	74.6

Insights from DC, and how to improve in future work

‼ Model capabilities

‼ Structured memories ➥ Project I

‼ Modularization ➥ **Project II**

In your projects will take the next steps:

➥ **Structured Memory**: KG construction, augmentation and refinements

- Improve retrieval and generalization

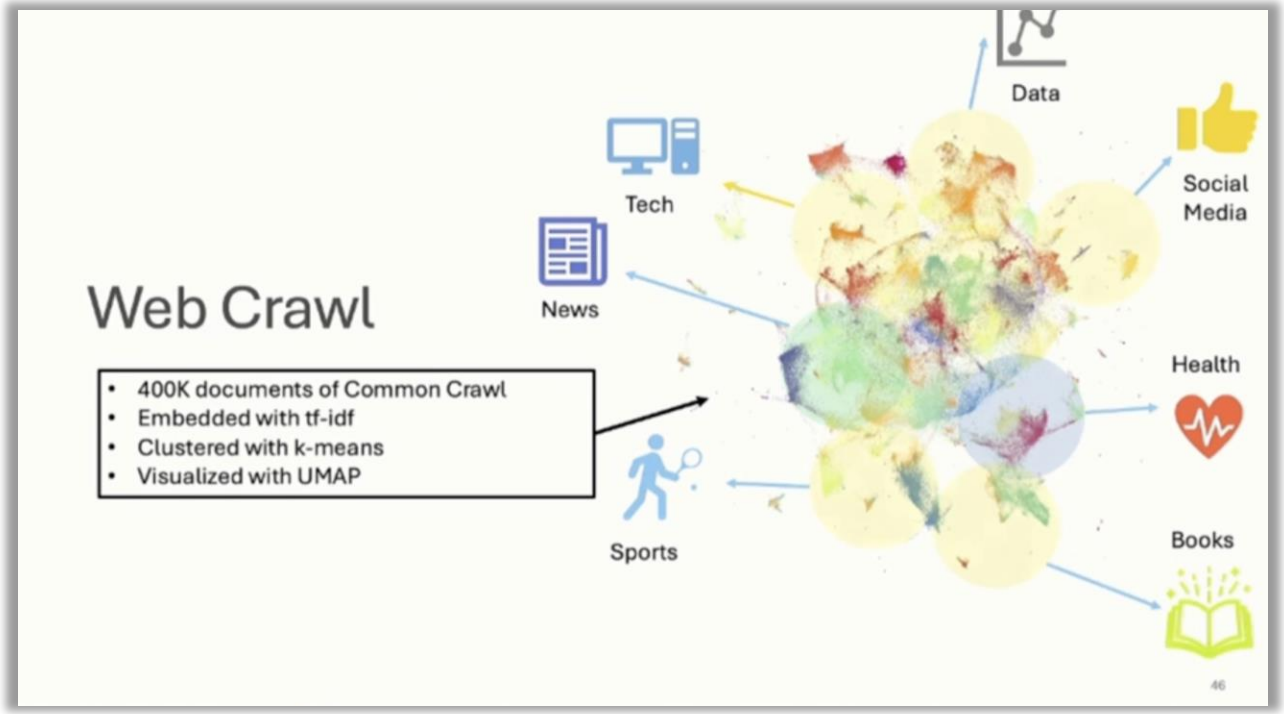
➥ **Modularized Memory?** Modularization using Mixture-of-Expert models

- Improve interpretability, domain or task specificity and performance

Modularization in Data

Ideally, we'd like to

- Select parts of a dataset that pertain to **specific domains** to
- Train **specialized models (experts)**

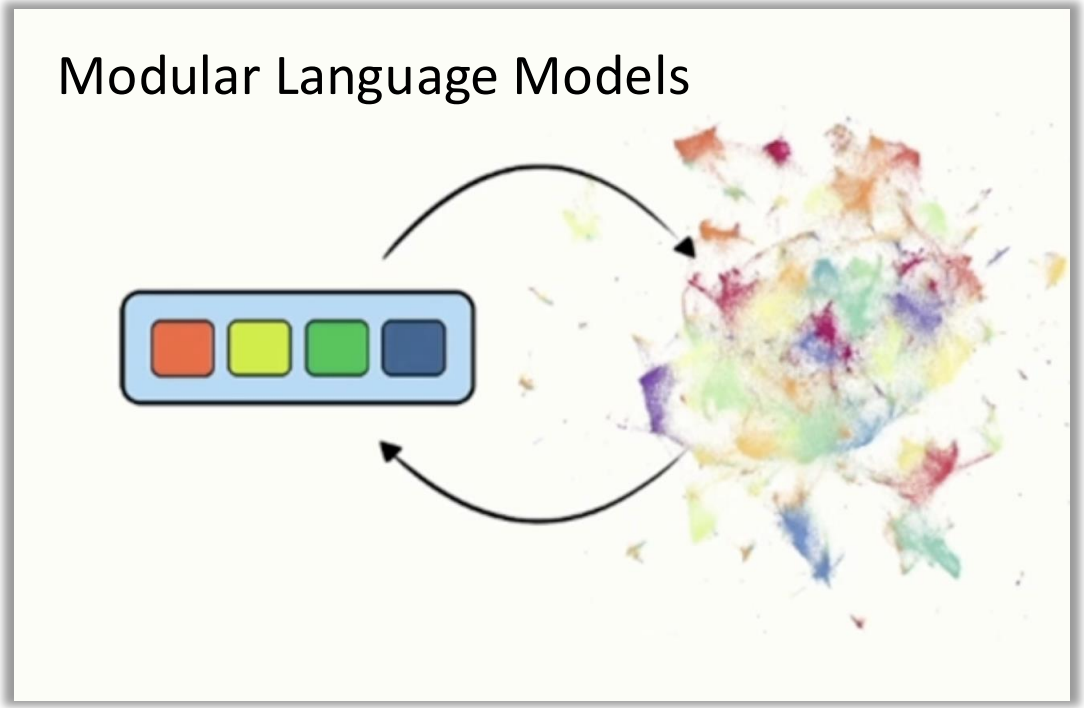


- For enhanced training **efficiency** and task **performance**

Modularization in Data and Models

Or, we could

- Specialize different parts of a model (**experts**) to different parts of the data (**domains and tasks**)



- Allowing us to **Mix**, **add** or **remove** experts as necessary

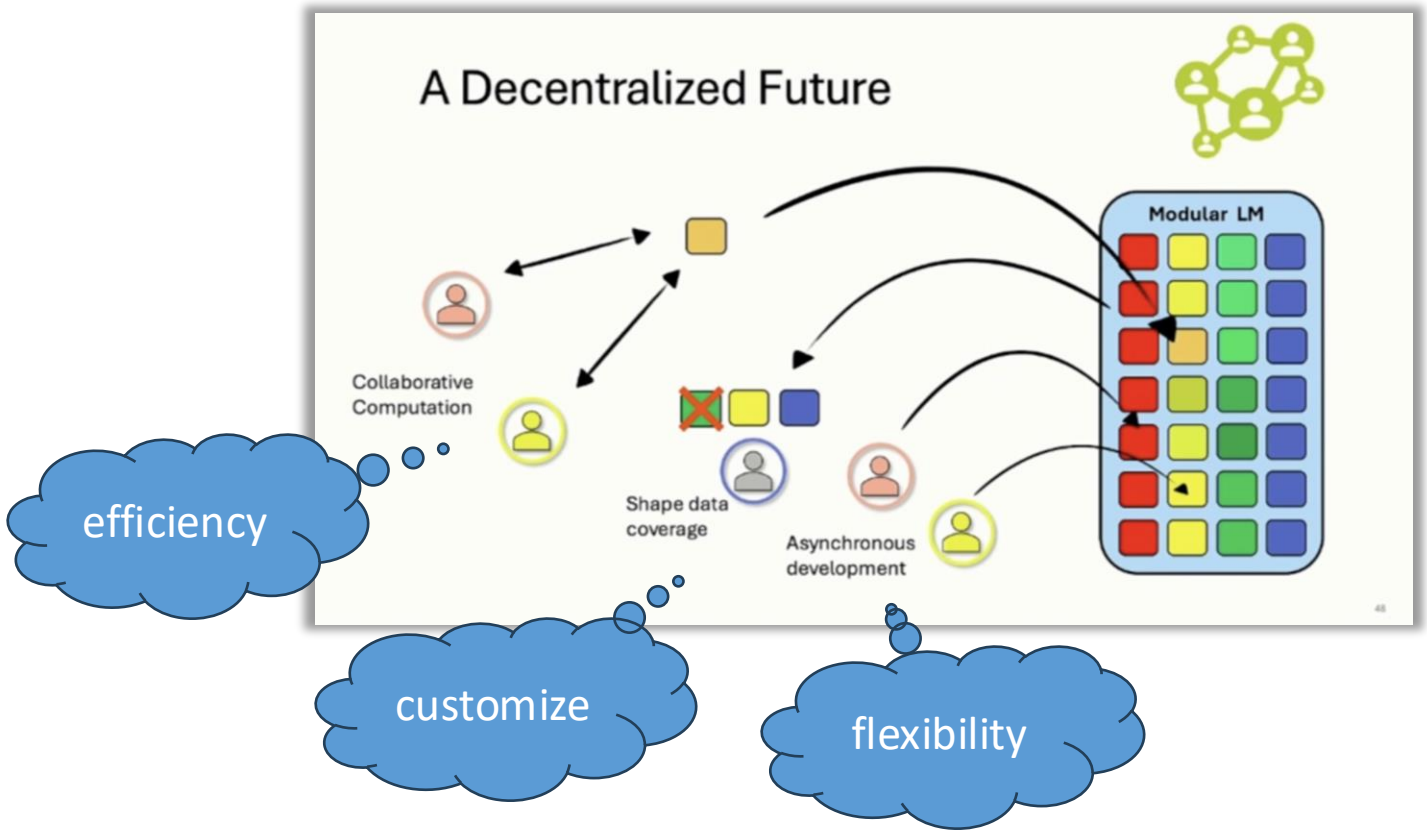
- To gain **flexibility** and **customization** at test time (+ enhance performance)

Modularization in Data and Models

Or, we could

- Specialize different parts of a model (**experts**) to different parts of the data (**domains**)

- Allowing us to **Mix**, **add** or **remove** experts as necessary

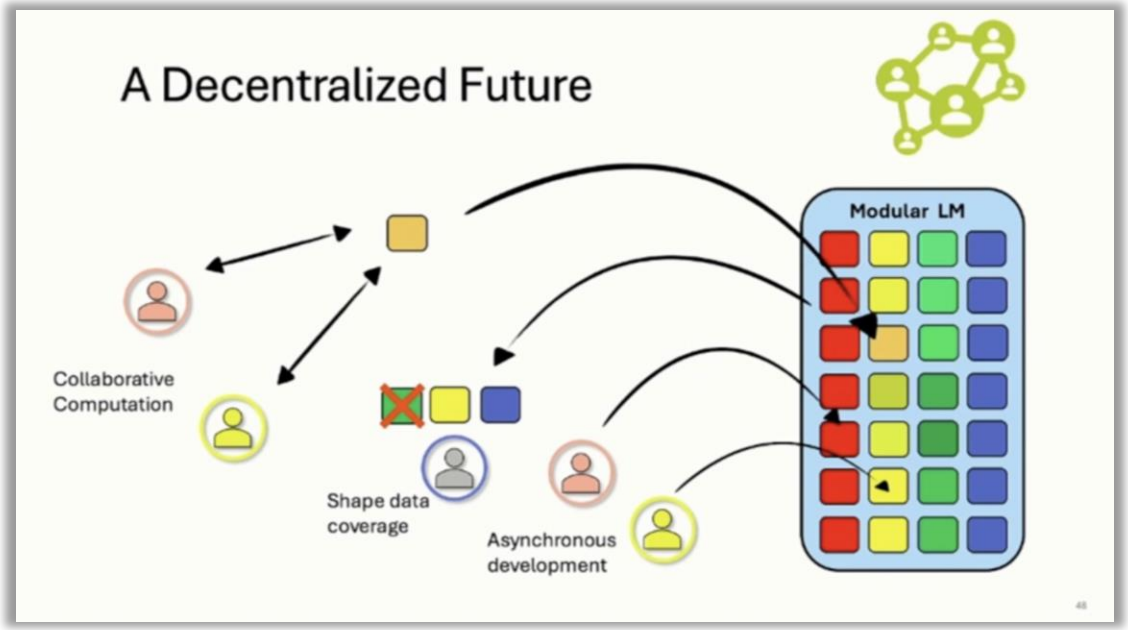


- To gain **flexibility** and **customization** at test time (+ enhance performance)

FlexOLMo

(Shi et al. 2025)

- **Modular:** Incorporates experts that can be cheaply customized **after training**.
- **Asynchronous**: Train and update different experts **independently** at arbitrary **scale**.
- **Sparse**: Despite training many experts, use only a **few experts** at inference time.



Mixture of Experts (MoE)

Shazeer et al. 2017

- FF layers contain several **expert modules**
- A **router** selects, for given inputs, specific modules through which it is passed.
- This way, a given module **specializes** for a given function.

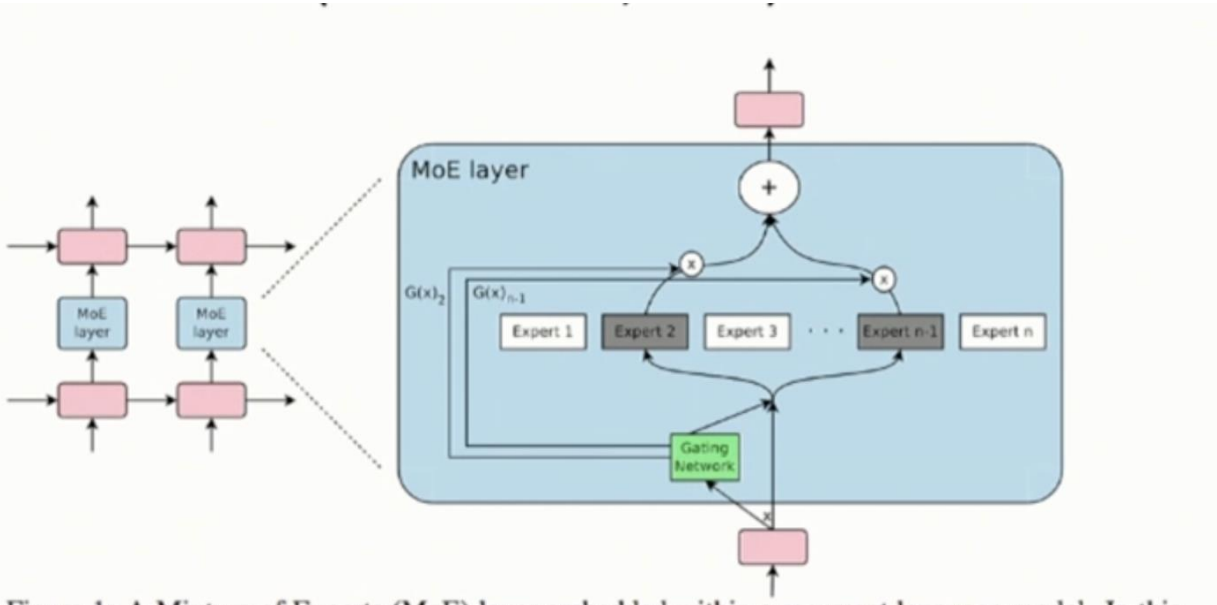


Figure 1: A Mixture of Experts (MoE) layer embedded within a recurrent language model. In this case, the sparse gating function selects two experts to perform computations. Their outputs are modulated by the outputs of the gating network.

For routing options see e.g.:

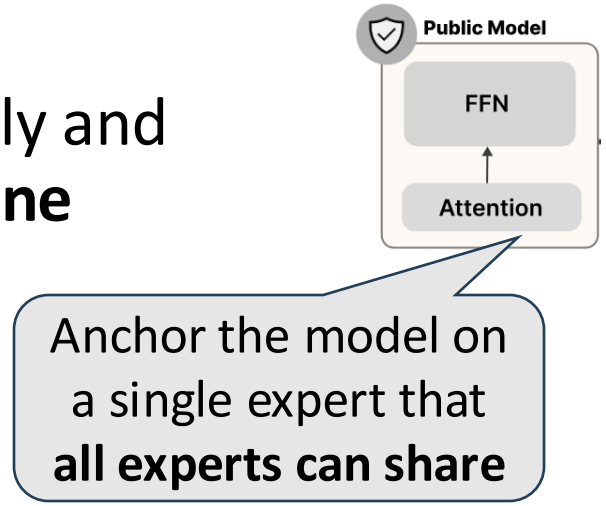
Modular Deep Learning, TMLR.

Mixture of Cognitive Reasoners: Modular Reasoning with Brain-Like Specialization

FlexOLMo

(Shi et al. 2025)

- Can experts be trained **independently** on different sources?
- How to efficiently and flexibly **recombine** the modules?



Goal 1: **independently** train experts on disjoint (possibly private) data
Goal 2: **add** and **remove** during inference with **no further training**

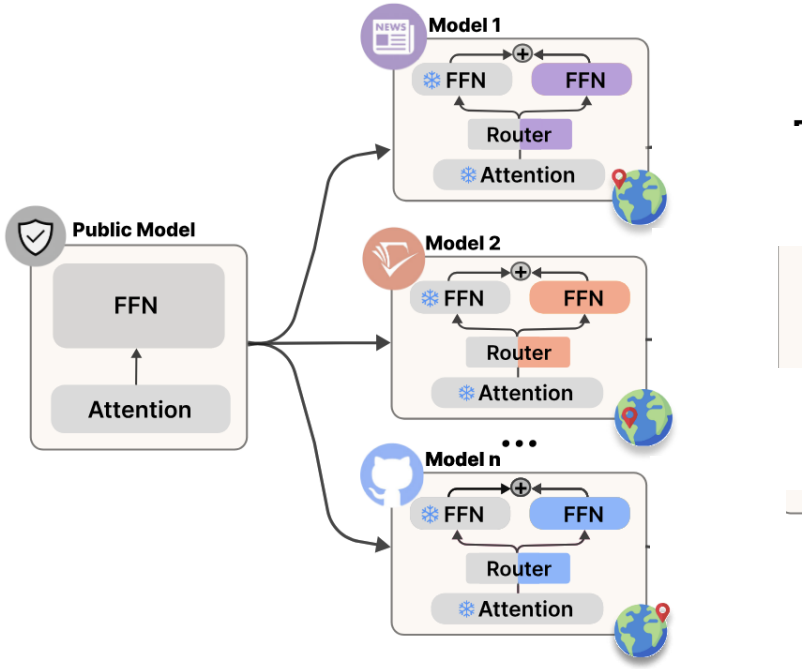
https://allenai.org/blog/flexolmo

FlexOLMo

(Shi et al. 2025)

- Can experts be trained **independently** on different sources?
- How to efficiently and flexibly **recombine** the modules?

Create modules



How to recombine?

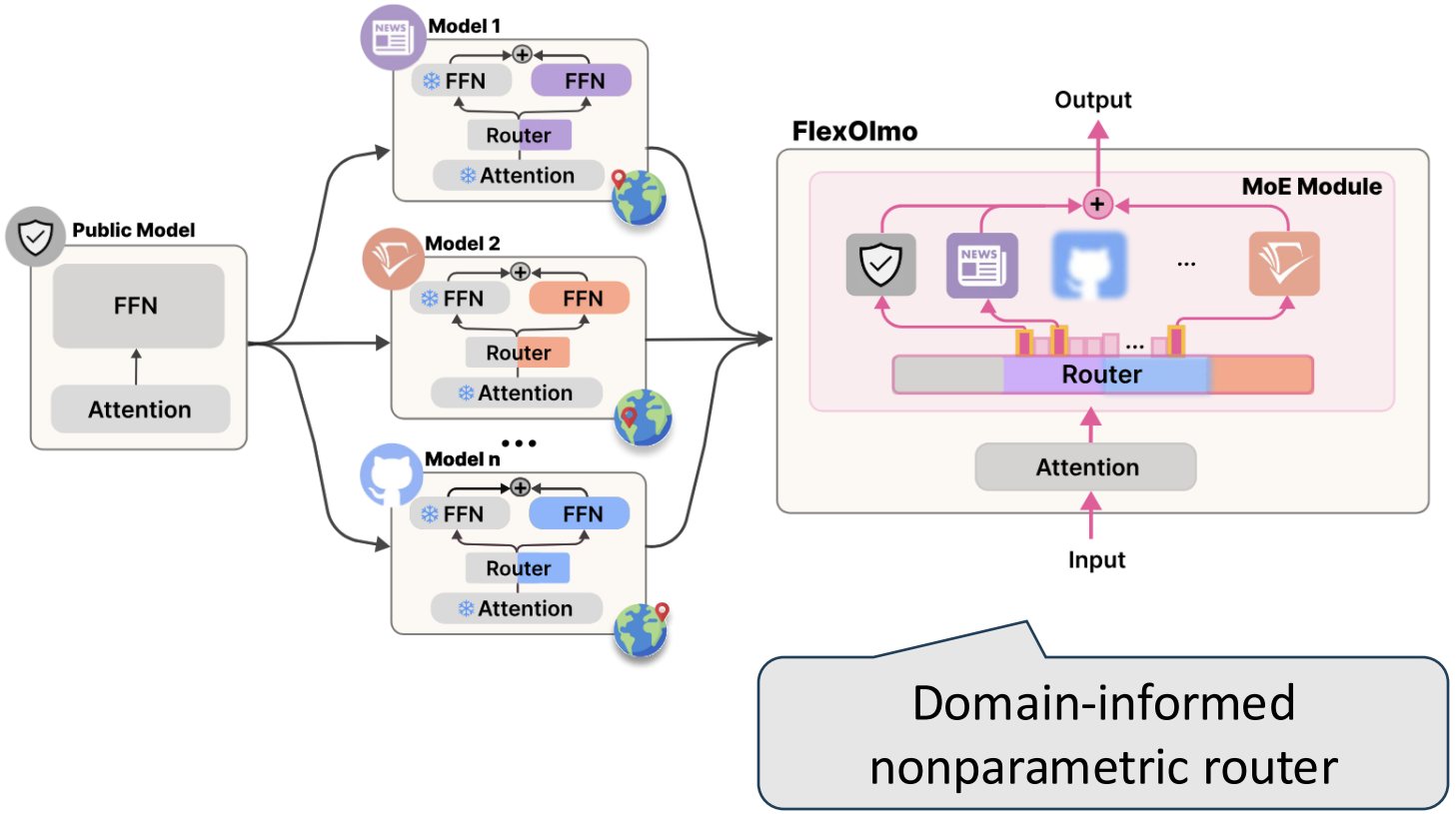
Goal 1: **independently** train experts on disjoint (possibly private) data
Goal 2: **add** and **remove** during inference with **no further training**

https://allenai.org/blog/flexolmo

FlexOLMo

(Shi et al. 2025)

- Can experts be trained **independently** on different sources?
- How to efficiently and flexibly **recombine** the modules?



https://allenai.org/blog/flexolmo

Routing as a classification problem

Training: *n pairwise binary classifiers*

C_{pub} *vs.* C_1

C_{pub} *vs.* C_2

….

C_{pub} *vs.* C_n

Inference: *a multi-class classifier*

$C_{pub}, C_1, C_2, …., C_n$

Routing as a classification problem

Training: *n pairwise binary classifiers*

~~C_{pub} vs. C_1~~ C_1 *vs. not* C_1

~~C_{pub} vs. C_2~~ C_2 *vs. not* C_2

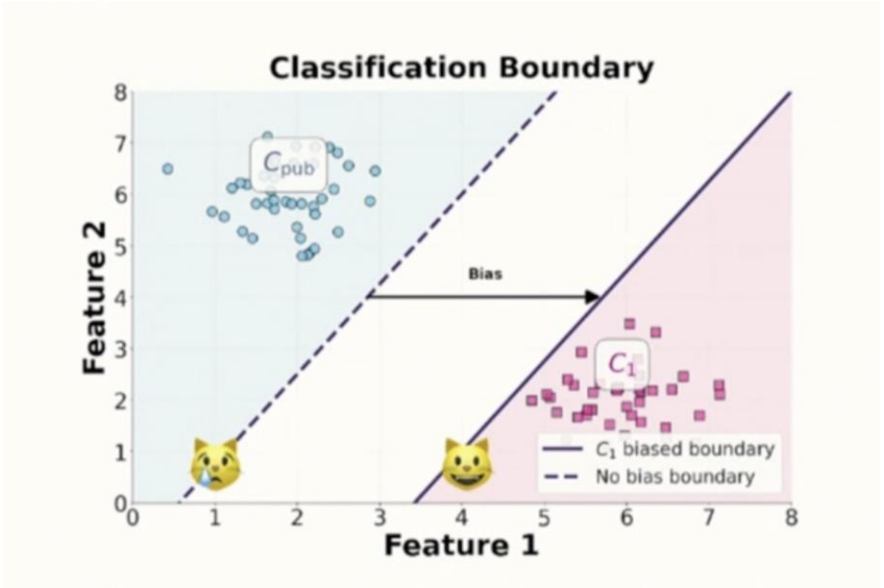
....

~~C_{pub} vs. C_n~~ C_n *vs. not* C_n

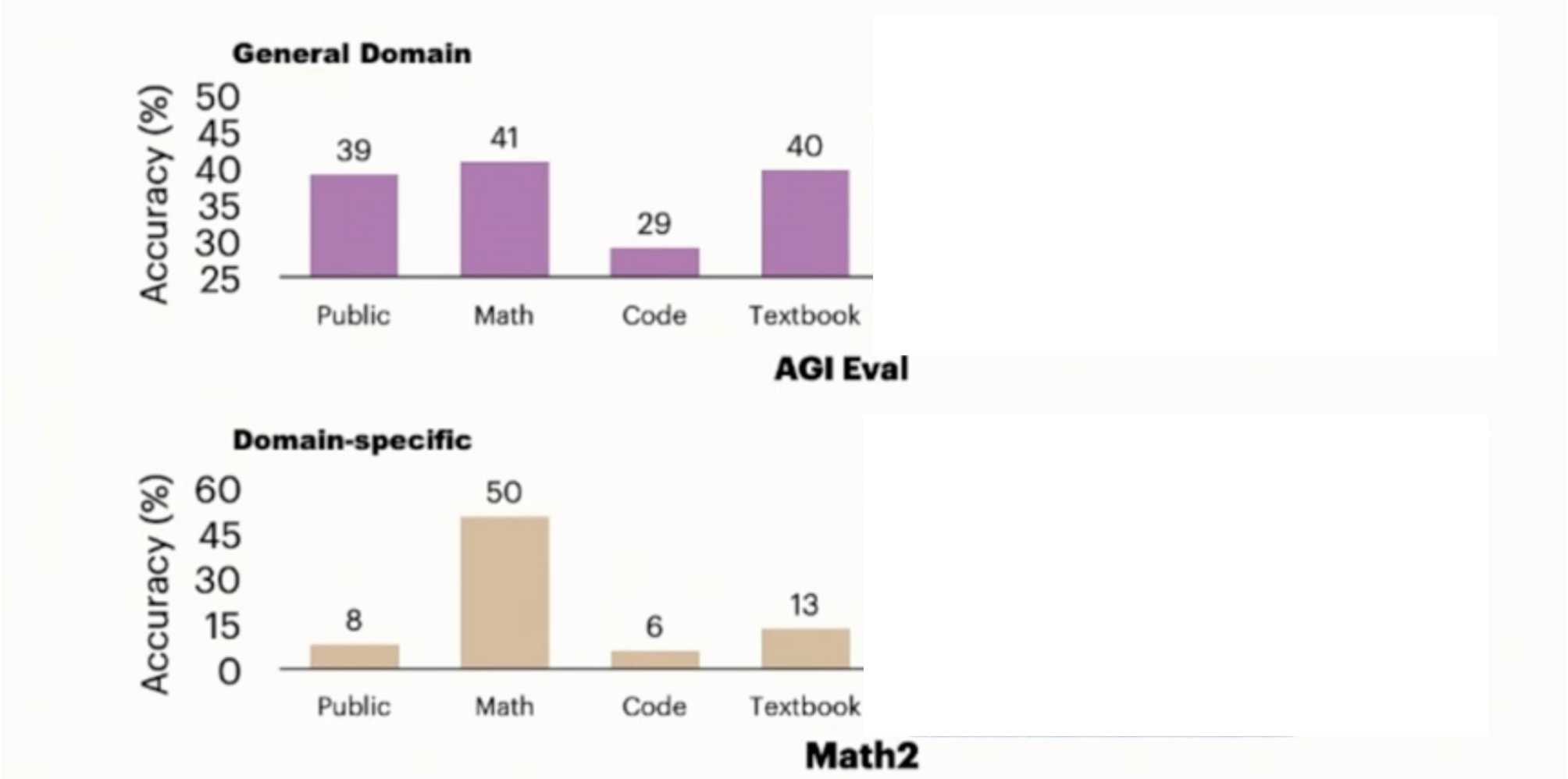
Idea: push **decision boundary** towards the **classes C_i** as much as possible!

Inference: *a multi-class classifier*

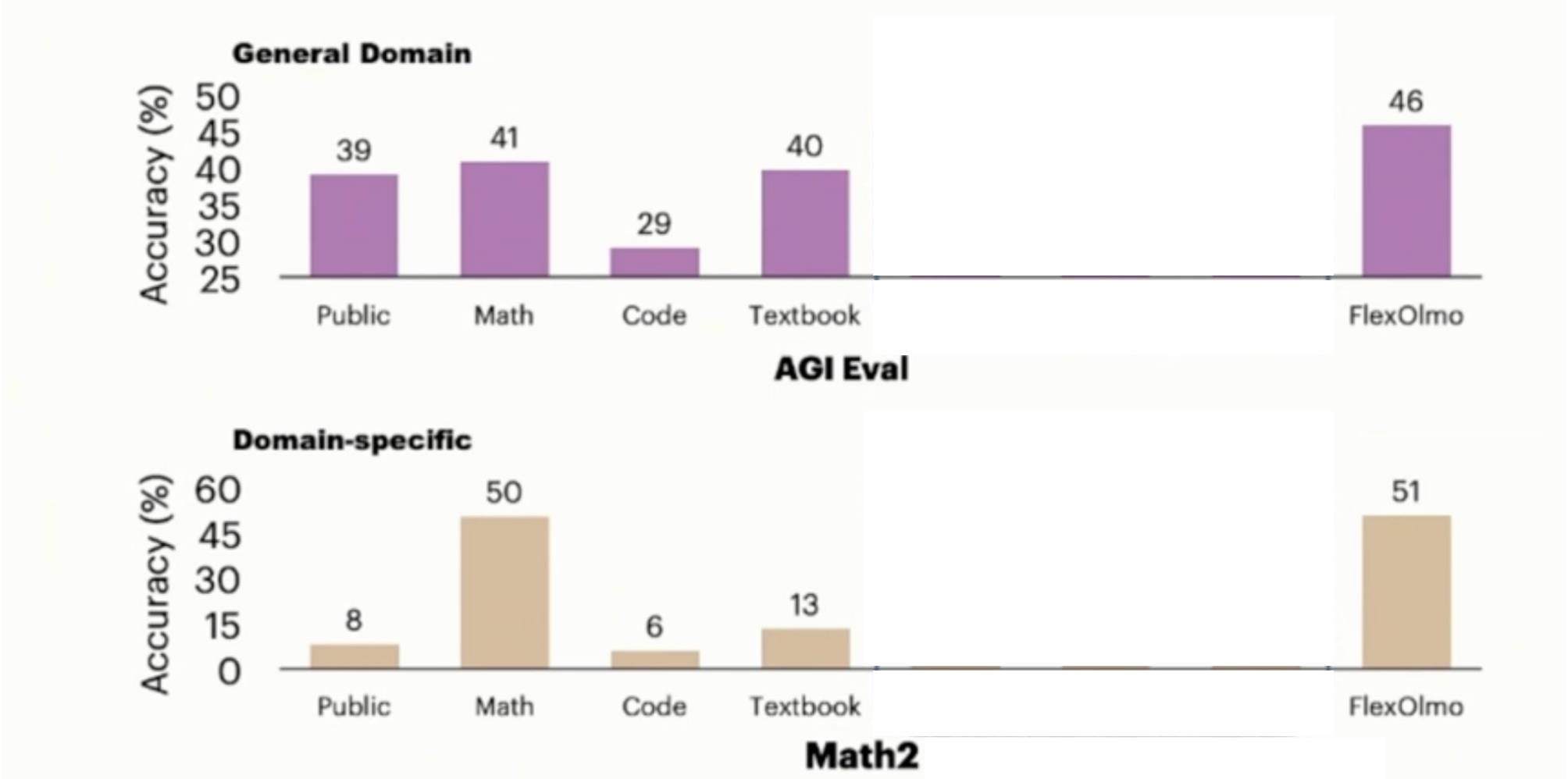
C_{pub}, C_1, C_2, …., C_n



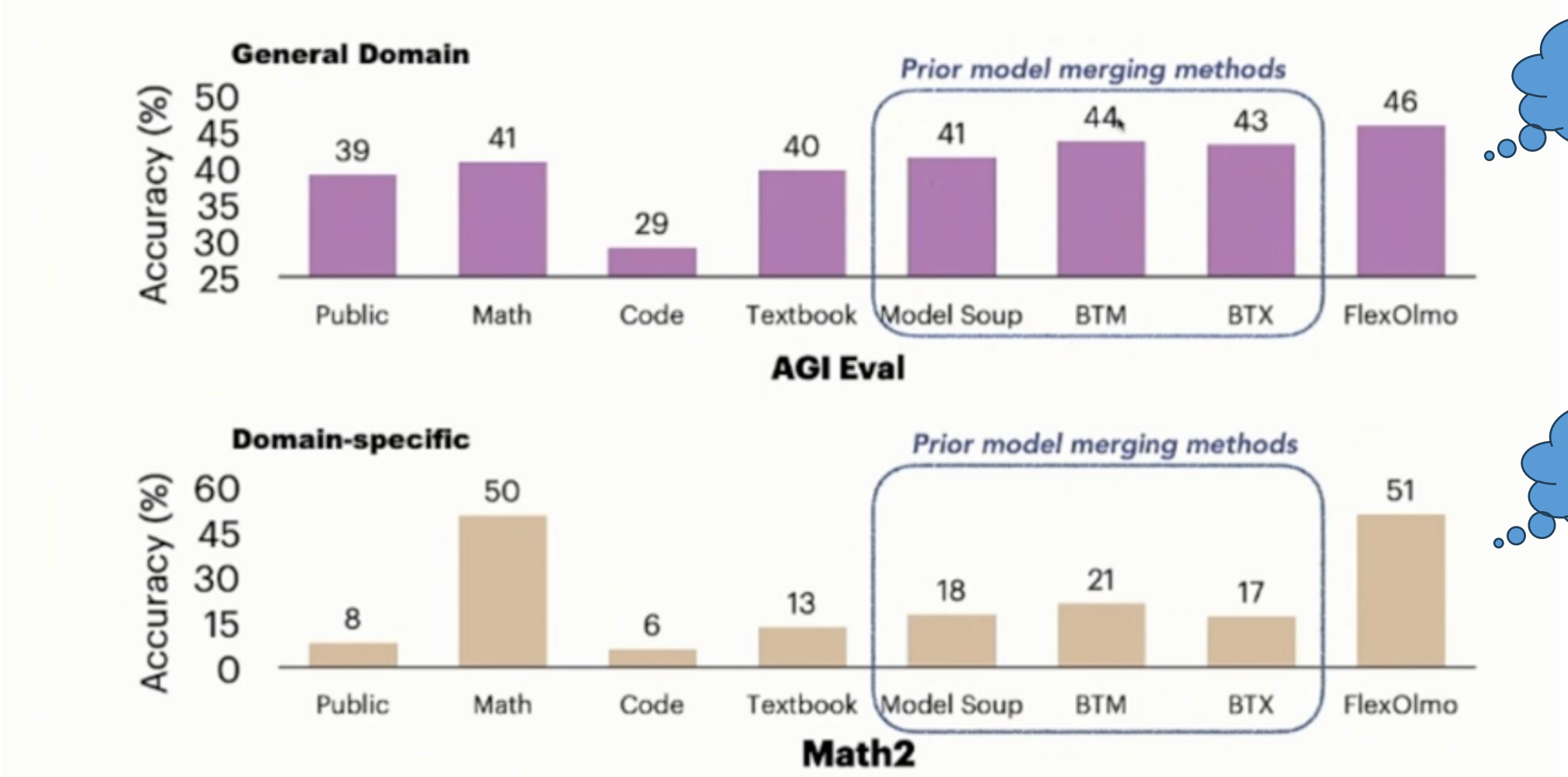
FlexOLMo: disentangling and merging experts



FlexOLMo: disentangling and merging experts



FlexOLMo: disentangling and merging experts

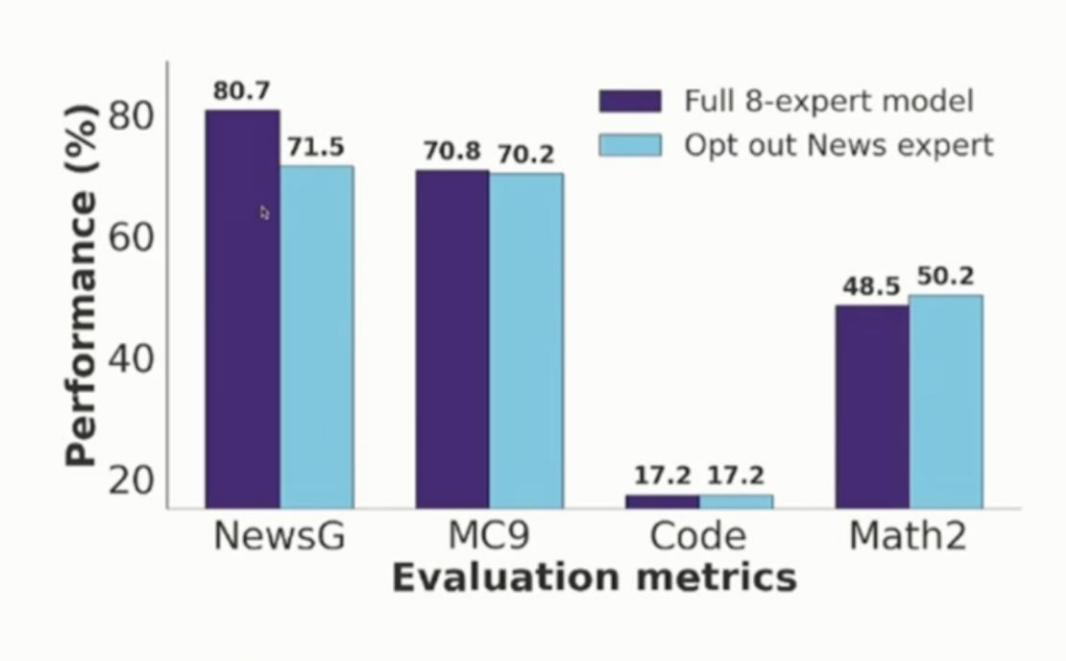


merging abilities

retaining abilities

FlexOLMo: Forging independent experts

- **Removing** an expert (here: News) weakens the targeted expertise (News) -- but not others.



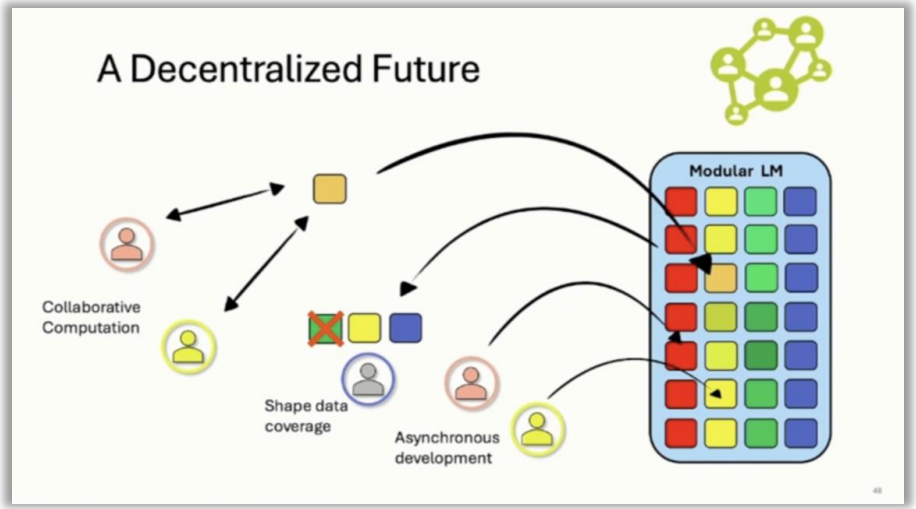
FlexOLMo

1. **Modular**, **distributed** model training with **independent datasets**
2. New routing scheme: freely **add** and **remove experts** during **inference**
3. Strong performance

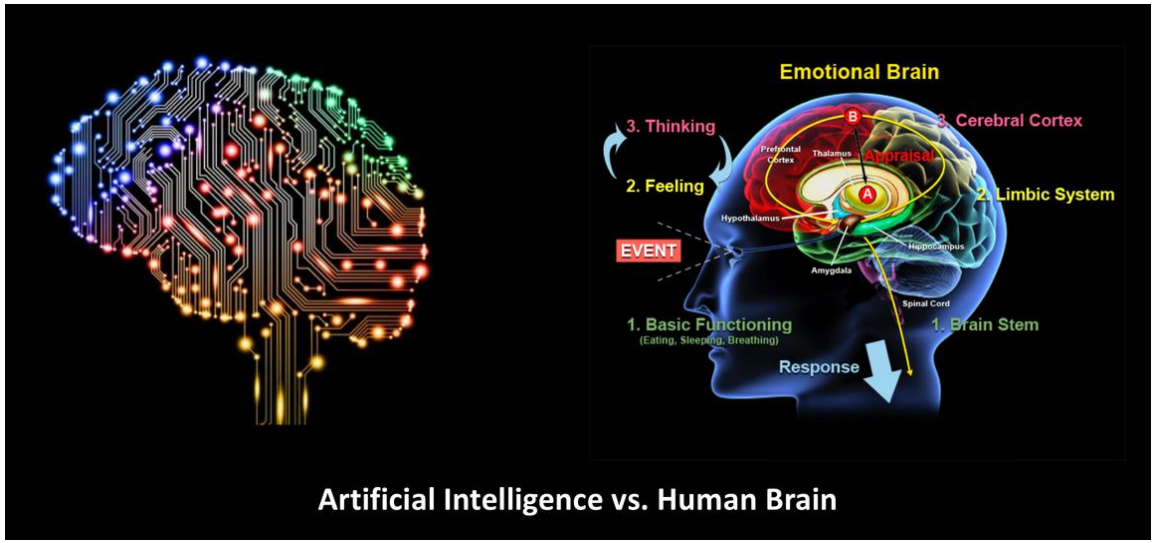
Model	hf.co/allenai/FlexOlmo-7x7B-1T
Code	github.com/allenai/FlexOlmo
Blog	allenai.org/blog/flexolmo

Project II: Modular Reasoning Models

- Can we create **memories for special knowledge domains**
  - for task specialization, better performance, flexibility and enhanced interpretability?



- Also: the **human brain** is organized in modules
- Are **modularized models** more **cognitively plausible** than standard LLMs?
  (cf. AlKhamissi et al. 2025)



https://www.linkedin.com/pulse/cognitive-chronicles-unraveling-intricacies-human-memory-mandal-/

Are LLMs cognitively plausible?

- The **Language Network** is connected to **other cognitive domains**
  - perception, motor planning, **knowledge and reasoning in**
  - multiple demand (MD) network, situation modelling, theory of mind reasoning (ToM)
- in processes of **language comprehension** and **production.**

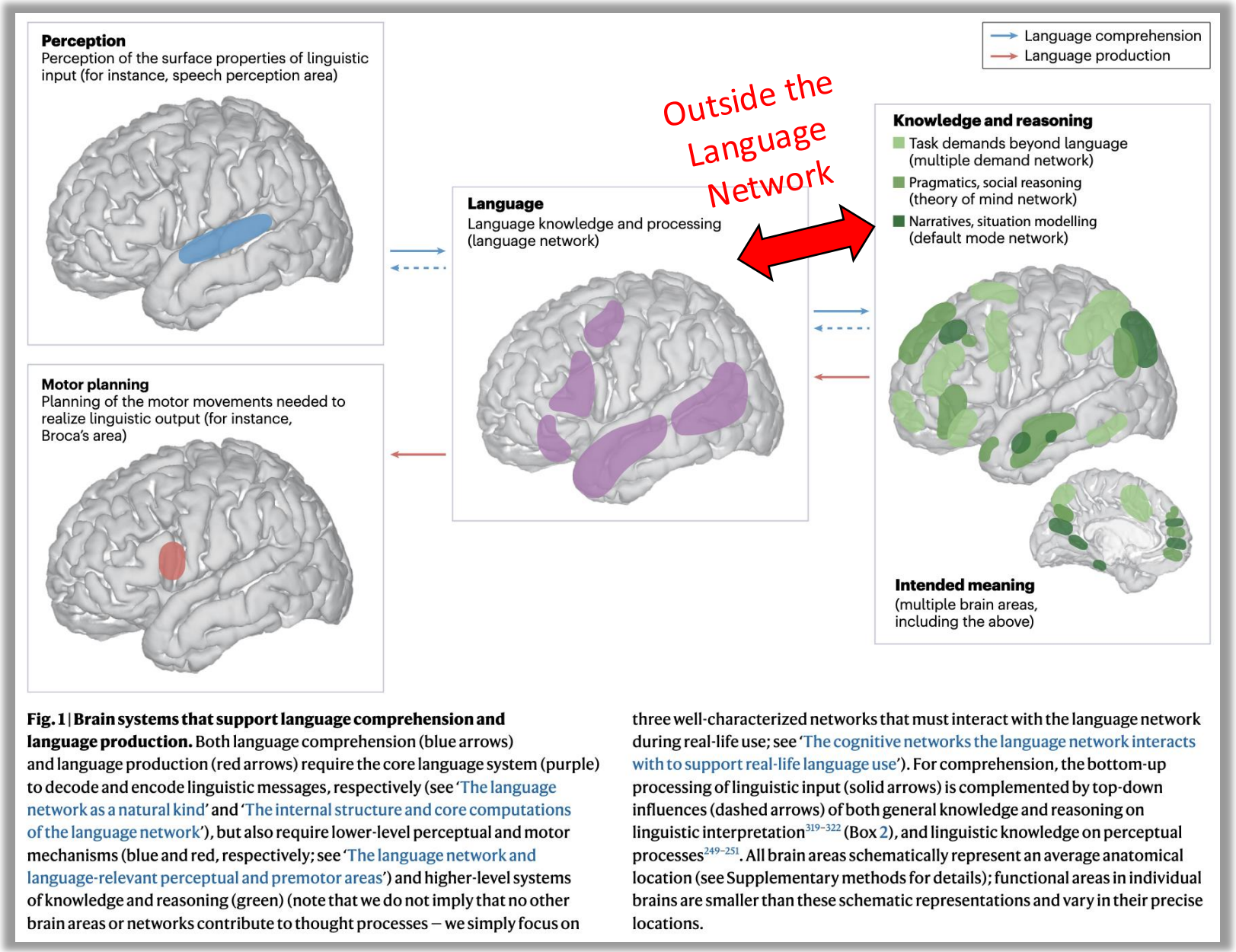


Fig. 1 | Brain systems that support language comprehension and language production. Both language comprehension (blue arrows) and language production (red arrows) require the core language system (purple) to decode and encode linguistic messages, respectively (see 'The language network as a natural kind' and 'The internal structure and core computations of the language network'), but also require lower-level perceptual and motor mechanisms (blue and red, respectively; see 'The language network and language-relevant perceptual and premotor areas') and higher-level systems of knowledge and reasoning (green) (note that we do not imply that no other brain areas or networks contribute to thought processes – we simply focus on three well-characterized networks that must interact with the language network during real-life use; see 'The cognitive networks the language network interacts with to support real-life language use'). For comprehension, the bottom-up processing of linguistic input (solid arrows) is complemented by top-down influences (dashed arrows) of both general knowledge and reasoning on linguistic interpretation[319–322] (Box 2), and linguistic knowledge on perceptual processes[249–251]. All brain areas schematically represent an average anatomical location (see Supplementary methods for details); functional areas in individual brains are smaller than these schematic representations and vary in their precise locations.

Fedorenko, E., Ivanova, A., Regev, T. (2024): The language network as a natural kind within the broader landscape of the human brain, Nature Reviews Neuroscience, 25, 289–312.

Project II: Modular Reasoning Models

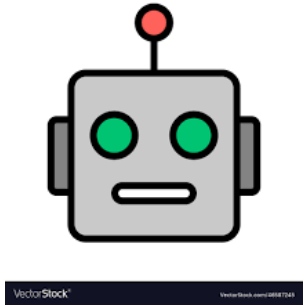
Anchor Model

- small-sized FlexOLMo (or other)

Module definition & induction

- **Routing**
  - Cluster KG relations to characterise domains
  - Domain classification w/ Linear/MLP classifier
    ➥ Adapters
- **Module Induction**
  - LoRA fine-tuning on
  - mixed-domain datasets

Modularization evaluation



Data Selection

- KGs and reasoning datasets

Defining routing function

- Relation sets based on
  - ConceptNet, ATOMIC, WebChild relation sets
  - Test on EWoK (11 domains)

evaluate & refine

Evaluation of modularization

- **Defined domains**
  - EWoK, ATOMIC, Soc.Chem. 101, ..
- **Mixed domain datasets**
  - CSQA, αNLI, SocialIQA, HellaSwag

Domain specific knowledge

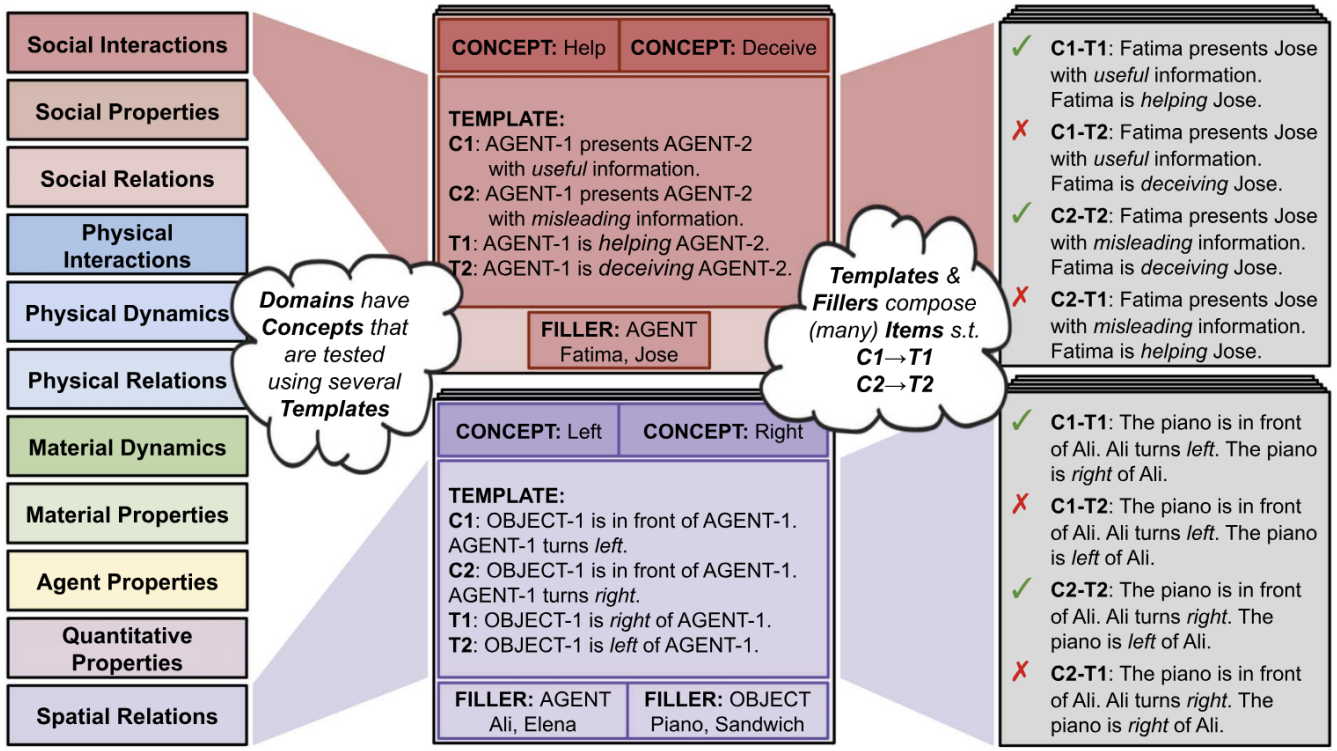


Figure 1: EWoK design, illustrated with examples from *social interactions & spatial relations*. Each *domain* contains a set of *concepts*, *contexts*, and *targets*. These combine to form many *templates*, which specify minimal pairs of contexts (C) and targets (T), such that T_1 matches C_1 but not C_2, and T_2 matches C_2 but not C_1. Each template can be combined with *fillers* to generate an even larger collection of *items*.

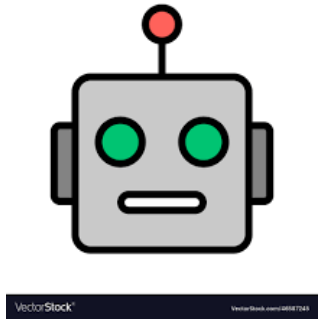
Elements of World Knowledge (EWOK): A Cognition-Inspired Framework for Evaluating Basic World Knowledge in Language Models, TACL 2025

Project II: Modular Reasoning Models

Anchor model:
- e.g. small FlexOLMo

Module induction:
- LoRA fine-tuning
- Linear ➥ MLP class. ➥ adapter

Routing:
- K clusters based on KG relation sets



Data: KGs and reasoning datasets

Define routing function
- EWoK (11 domains)
- ConceptNet, ATOMIC, WebChild

Evaluation of modularization
- EWoK, CSQA, α-NLI, SocialIQA

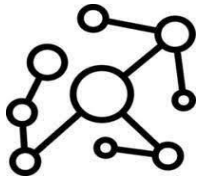
Module induction:
- Fine-tune FlexOlmo on mixed-domain task datasets
  - using LORA: linear classifier ➥ MLP ➥ adapters
  - initial refinement on EWoK (11 domains)
- Modularization evaluation
  - Mixed-domain datasets: CSQA, EWoK, α-NLI,HellaSwag, SocialIQA
  - Perform ablation tests and dependencies beween clusters
- Variations: textual vs. structured knowledge representation (➥ Project I)

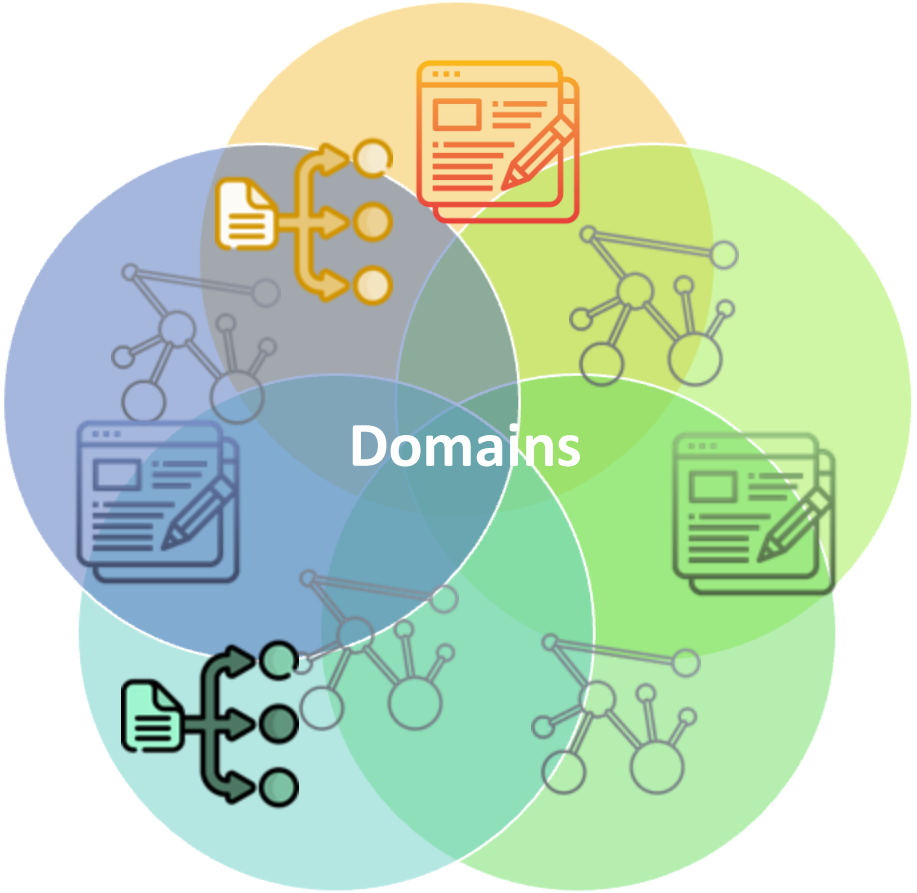
Dimensions of Modularization

Analyzing Distributions & Dependencies

Primary: Knowledge domains

Secondary:

- Knowledge resources 
- Reasoning tasks 
- Textual sources 



Evaluation

- Evaluation of Modularization
  - Domain-characteristic relation sets
  - Module interferences & dependencies
- Performance effects
  - +/- modularization
  - modularization variants
- Performance analysis
  - Modularization-based error analysis

Outcomes

- Modularizing datasets (and KGs): subdivide by domains
  - CSQA, HellaSwag, α-NLI
  - WinoGrande, Narrative Similarity
- Performance comparisons
  - Mixed- vs. selected or ablated domain
  - +/- modularization
  - modularization variants (dependencies, hierarchical, …)
- Interpretability & performance enhancement w/ adaptive memory

Recommendations

Use small anchor model

- FlexOLMo (small, 1B) or other

Start with Hugging Face PEFT (Parameter-Efficient Fine-Tuning) **+ LoRA**

- Mature, well-documented https://huggingface.co/docs/peft/.
- Simple to implement a custom router (top-1 or 2 predictions)
- First test linear or FF classifier on few modules
- Then extend to adapters

Adapters

- Primary Package Hugging Face PEFT: Supports *adapter-based methods*
  - adds extra trainable parameters after attention
  - to fully-connected layers of a frozen pretrained model
  - ➥ reduces memory usage, speeds up training

Implementation Aspects

Router: *Load-balanced routing*

- To prevent router collapse (when all inputs go to one expert), use auxiliary loss (see Switch Transformers, Fedus et al. 2022)
- Proceed from *top-k routing* to *learned mixture weights*.

Load balancing: Balance computational load across experts in training.

- Techniques: *importance weighting*, *expert dropout*, or *explicit load constraints*.

Evaluate for modularization:

- Track per-expert performance on different domains, find routing patterns, measure loss per expert to confirm differentiation.

Data:

- Use multi-domain training data to encourage specialization.
  Use topical words or domain labels during training to guide expert development

Fedus et al. 2022: Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity, JMLR.

Try a staged approach

1. Discovery: Start with **Linear Router**

- Minimum complexity, gives insight into the data
- Examine learned weights: which input features drive which adapters?

1.5 Validate: Switch to **Task/Domain-Aware Router**

- Condition on EWoK ground-truth domains
- Evaluate router predictions on EWoK labels to validate modularization effects

2. Refinement: Move to **Attention-based or MLP Router**

- Especially if linear/task-aware router is insufficient
- Use attention to trace *which aspects* of the input drive adapter selection
- ➥ supports analysis: which source features activate which adapters?

3. Scaling: Try a **Load-Balanced MoE Router:** Essential for scaling up

References

- Wu, Y., Liang, S., Zhang, C., Wang, Y., Zhang, Y., Guo, H., Tang, R., Liu, Y. (2025): From Human Memory to AI Memory: A Survey on Memory Mechanisms in the Era of LLMs, arXiv.
- Suzgun, M., Yuksekgonul, M., Bianchi F., Jurafsky, D., Zo, J. (2025): **Dynamic Cheatsheet: Test-Time Learning with Adaptive Memory**, arXiv, [GitHub]
- Kim, J., Rhee, S., Kim, M., Kim, D., Lee, S., Sung, Y., Jung, K. (2025): ReflAct: World-Grounded Decision Making in LLM Agents via Goal-State Reflection , accepted for EMNLP. (no code) [Summary: Chatpaper]
- Can LLMs be Good Graph Judge for Knowledge Graph Construction? EMNLP.

Models

- Plenz, M. Heinisch, P., Cimiano, P., Frank, A (2023): Similarity-weighted Construction of Contextualized Commonsense Knowledge Graphs for Knowledge-intense Argumentation Tasks. ACL.
- Plenz, M., Frank, A. (2024): Graph Language Models. ACL .
- Shi, W., Bhagia, A., Farhat, K., Muennighoff, .. (2025): **FLEXOLMO: Open Language Models for Flexible Data Use**, arXiv.
- Pfeiffer, J., Ruder, S., Vulic, I., Ponti E. M. 2023: Modular Deep Learning, TMLR.

Cognition

- AlKhamissi, B., De Sabbata, CN, Chen, Z., Schrimpf, M., Bosselut, A. (2025): Mixture of Cognitive Reasoners: Modular Reasoning with Brain-Like Specialization, arXiv.
- Fedorenko, E., Ivanova, A., Regev, T. (2024): The language network as a natural kind within the broader landscape of the human brain, Nature Reviews Neuroscience, 25, 289–312.

Task Datasets

- Yu, W. et al. (2020): ReClor: A Reading Comprehension dataset requiring logical reasoning. ICLR.
- Xu, F. et al. (2024): Are Large Language Models Really Good Logical Reasoners?. arXiv. (contains further datasets)
- Bhagavatula, C., et al. (2020): Abductive Commonsense Reasoning. ICLR. (αNLI)
- Ivanova, A. et al. (2025): Elements of World Knowledge (EWOK): A cognition-inspired framework for evaluating basic world knowledge in language models. arXiv, to appear in TACL.
- Speer. R., Chin, J., Havasi, C. (2017): ConceptNet 5.5: an open multilingual graph of general knowledge. AAAI.
- Tandon, N., de Melo, G., Weikum, G. 2017. WebChild 2.0: Fine-Grained Commonsense Knowledge Distillation, ACL.
- Hwang et al. 2021: (COMET-)ATOMIC2020: On Symbolic and Neural Commonsense Knowledge Graphs, AAAI.
- Talmor, A., Herzig, J., Lourie, N., Berant, J. 2019. CommonsenseQA: A Question Answering Challenge Targeting Commonsense Knowledge, NAACL.
- Zellers, R, Holtzman, A, Bisk, Y. Farhadi, A, Choi, Y. 2019: HellaSwag: Can a Machine Really Finish Your Sentence? ACL
- Chizhov, P., Nee, Langlais, P., 2025. What the HellaSwag? On the Validity of CS Reasoning Benchmarks. arXiv.
- Saha, S. 2021: ExplaGraphs: An Explanation Graph Generation Task for Structured Commonsense Reasoning. EMNLP
- Kazemi, M., et al. (2025): BIG-Bench Extra Hard. arXiv.
- Hatzel, H.O., et al. (2025): SemEval-2026 Task 4: Narrative Story Similarity and Narrative Representation Learning.
- Sap, M. et al. (2019): SocialIQA: Commonsense Reasoning about Social Interactions. EMNLP.
- A curated list of the most important common-sense datasets in NLP

Questions?

Make your Choice.

Project I and II can both accommodate 5 people