Planning in the Dark: LLM-Symbolic Planning Pipeline **Without Experts**

Sukai Huang,
Nir Lipovetzky,
and Trevor Cohn

THE UNIVERSITY OF MELBOURNE

1. Limitations in Existing Pipeline 😭

Fragile Pipeline: LLM-generated PDDL fail >99.9% of the time—requires ***expert***!

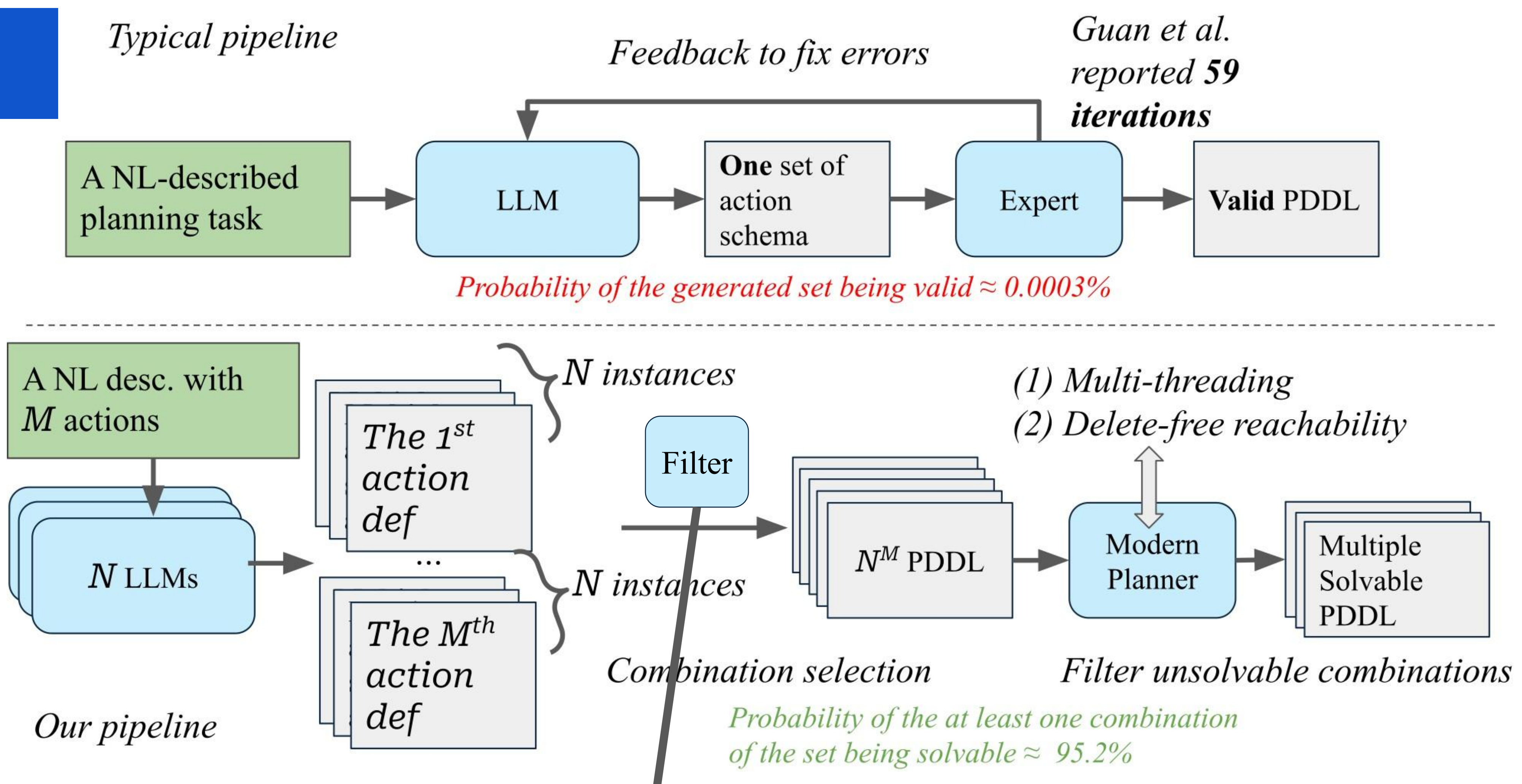
Expert Bottleneck & Bias: Heavy expert refinement (about. ***59 iterations***) + single-perspective bias

2. Solvable Schemas: A Simple Fix! 🤔

Multiple LLMs + Inter Schema Set combination: the probability of ***not*** finding a solvable set becomes $(1-p^M)^{N^M} \rightarrow 0$ where N is #LLMs, M is #actions, p is the prob. of valid action schema (single LLM)

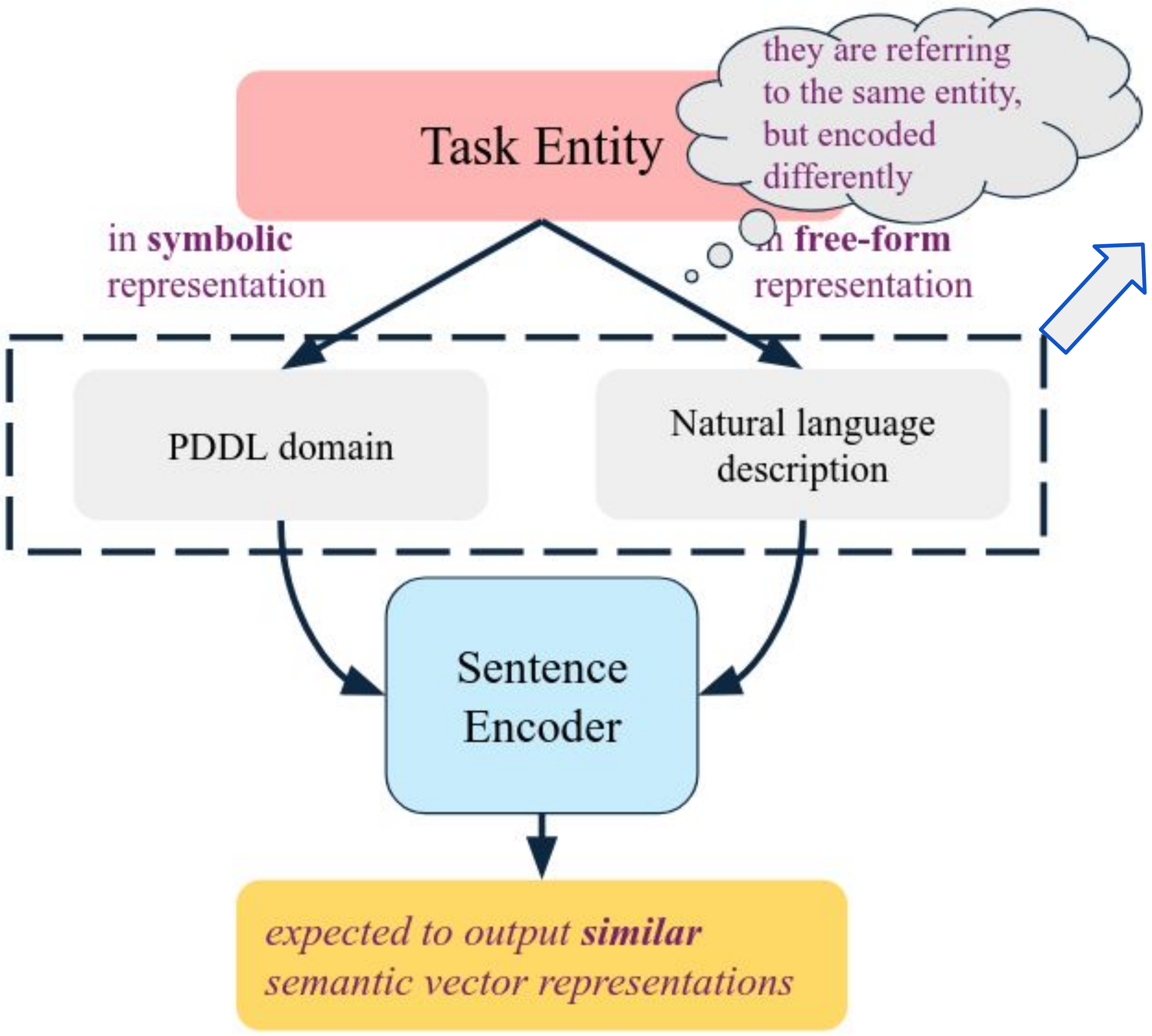
Adv: Solvable Shema Without Experts!

Disadv: brute force, semantic misalign


Typical pipeline
Feedback to fix errors
Guan et al. reported **59 iterations**
A NL-described planning task
LLM
One set of action schema
Expert
Valid PDDL
Probability of the generated set being valid ≈ 0.0003%
A NL desc. with M actions
N LLMs
The 1st action def
...
The Mth action def
N instances
N instances
Filter
Combination selection
N^M PDDL
(1) Multi-threading
(2) Delete-free reachability
Modern Planner
Multiple Solvable PDDL
Filter unsolvable combinations
Probability of the at least one combination of the set being solvable ≈ 95.2%
Our pipeline

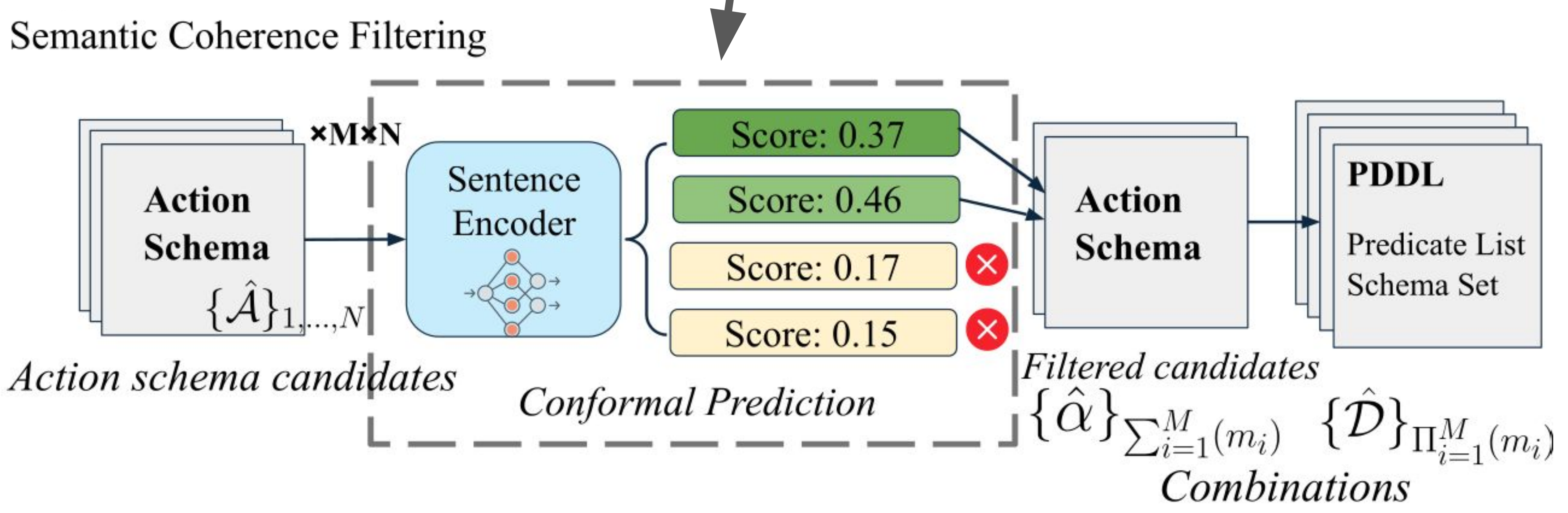
3. Weaver (1952)'s assumption 🤩

Like translation, there is a "common base of meaning" between natural language task and symbolic schemas.


Task Entity
they are referring to the same entity, but encoded differently
in **symbolic** representation
in **free-form** representation
PDDL domain
Natural language description
Sentence Encoder
expected to output **similar** semantic vector representations

Concept from the father of machine translation, Warren Weaver "Translation" (1952)

4. Filtering and Ranking Inspired by Weaver (1952)



Semantic score: **Schema Filter** and ***even*** **Plan ranking!**

5. Fine-tuning the Sentence Encoder is ***Convenient***!

Contrastive training with hard negatives synthesized via precon & effect manipulation

Manipulation Type	Description	Example
Swap	Exchanges a predicate between preconditions and effects	Precondition: `(at ?x ?y)` Effect: `(not (at ?x ?z))` → Precondition: `(not (at ?x ?z))` Effect: `(at ?x ?y)`
Negation	Negates a predicate in either preconditions or effects	Precondition: `(clear ?x)` → Precondition: `(not (clear ?x))`
Removal	Removes a predicate from either preconditions or effects	Precondition: `(and (on ?x ?y) (clear ?x))` → Precondition: `(on ?x ?y)`
Addition	Adds mutually exclusive (mutex) predicates to preconditions or effects (Helmert 2009]	Effect: `(on-table ?x)` → Effect: `(and (on-table ?x) (holding ?x))`

6. Contributions & find out more 👍

1. Address NL ***ambiguity*** by having ***diverse interpretation*** of the schema
2. Semantic validation, filtering and ranking ***without experts*** (2 min avg per problem for a 32-thread CPU, faster than expert-in-the-loop pipeline; 10 LLMs are adequate for ~8-action problems)
3. In fact, the proposed pipeline also allows ***lightweight*** expert intervention to further enhance accuracy too! Find our paper to see the details!