LLM

Victor Guichard, Eliott Vigier, Paul Chu

Introduction

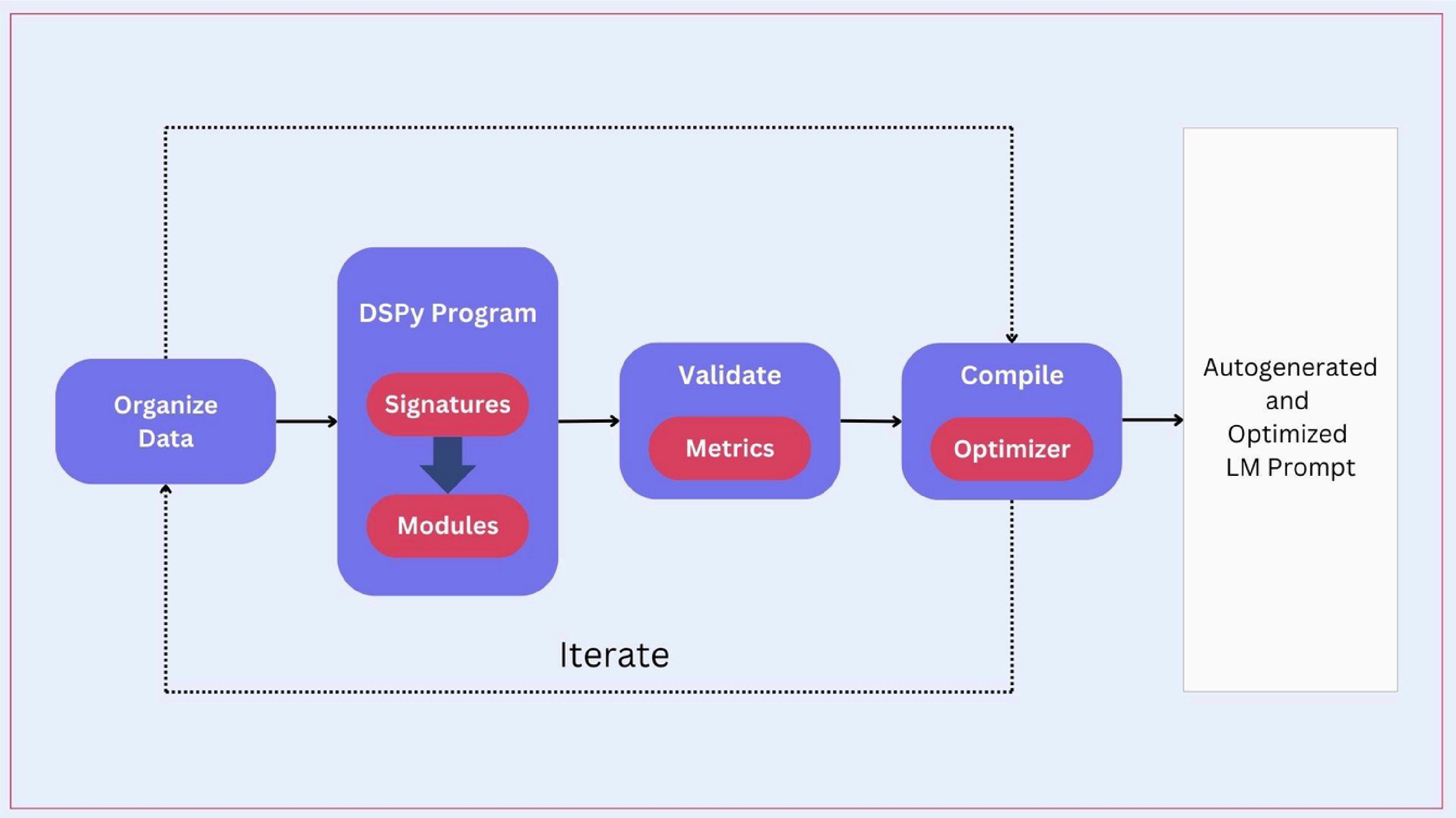
DSPy

Plan

- Introduction sur DSPy
- Reproduction et étude de DSPy
- LLM Ensembling
- Conclusion

Introduction sur DSPy

Declarative Self-improving Prompting in Python



DSPy affine le prompt du LLM de façon itérative afin de maximiser le score du LLM sur une tâche.

Introduction sur DSPy

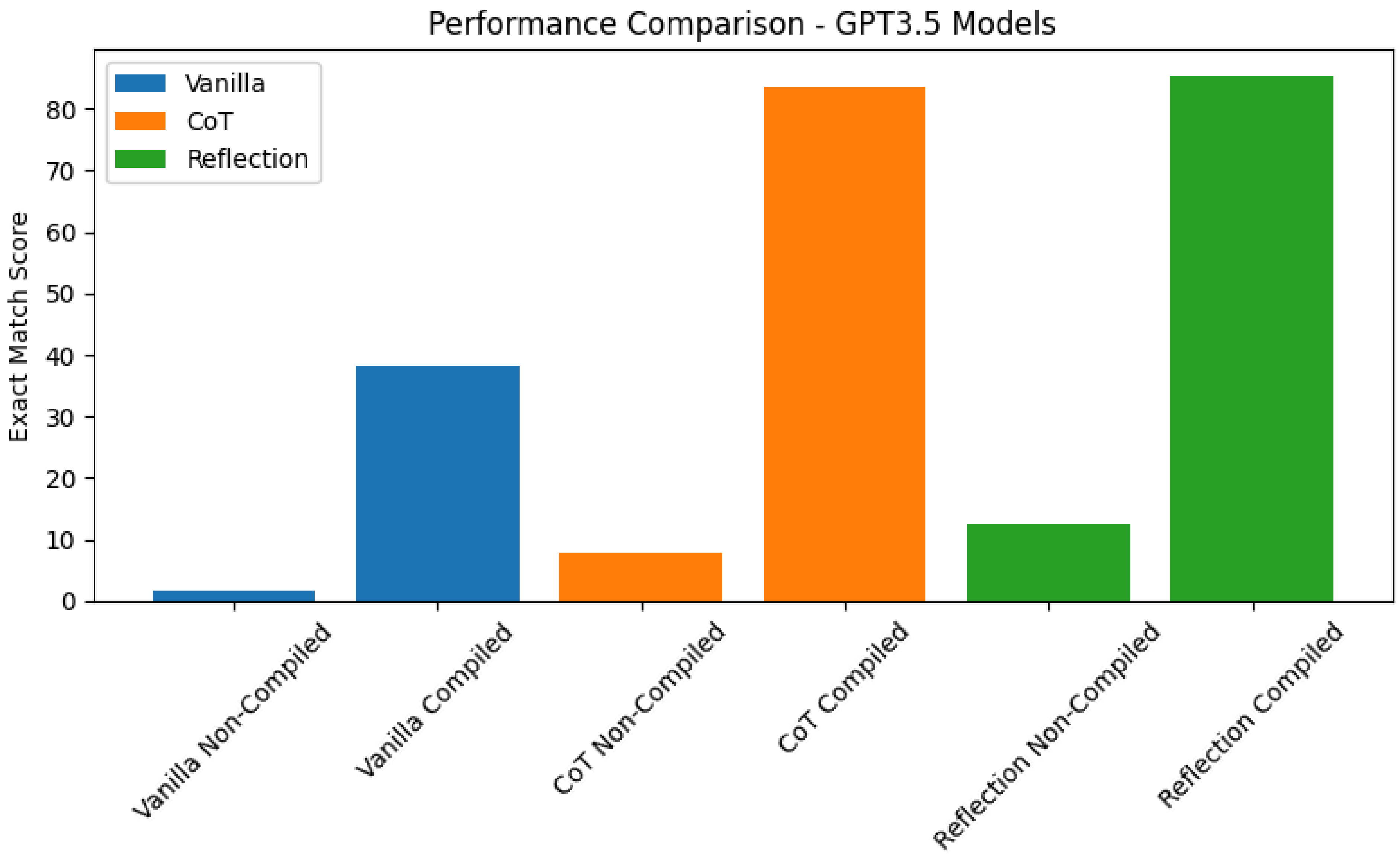
Signature : structure de la tâche à réaliser.
Ex : GSM8k, Questions (str) → Réponses (str)
Liée à un **module**.

Module : Stratégie de raisonnement.
Ex : Vanilla, Chain-of-Thought (CoT), Reflective.

Optimizer : Génère plusieurs prompts grâce au dataset. Compare les réponses de ces prompts avec la vraie réponse grâce à la **métrique** pour tout le dataset et garde le meilleur prompt.
Ex: bootstrap fewshot

Métrique : Compare le résultat du prompt généré avec la réponse attendue.
Ex: ExactMatch

Reproduction de DSPy



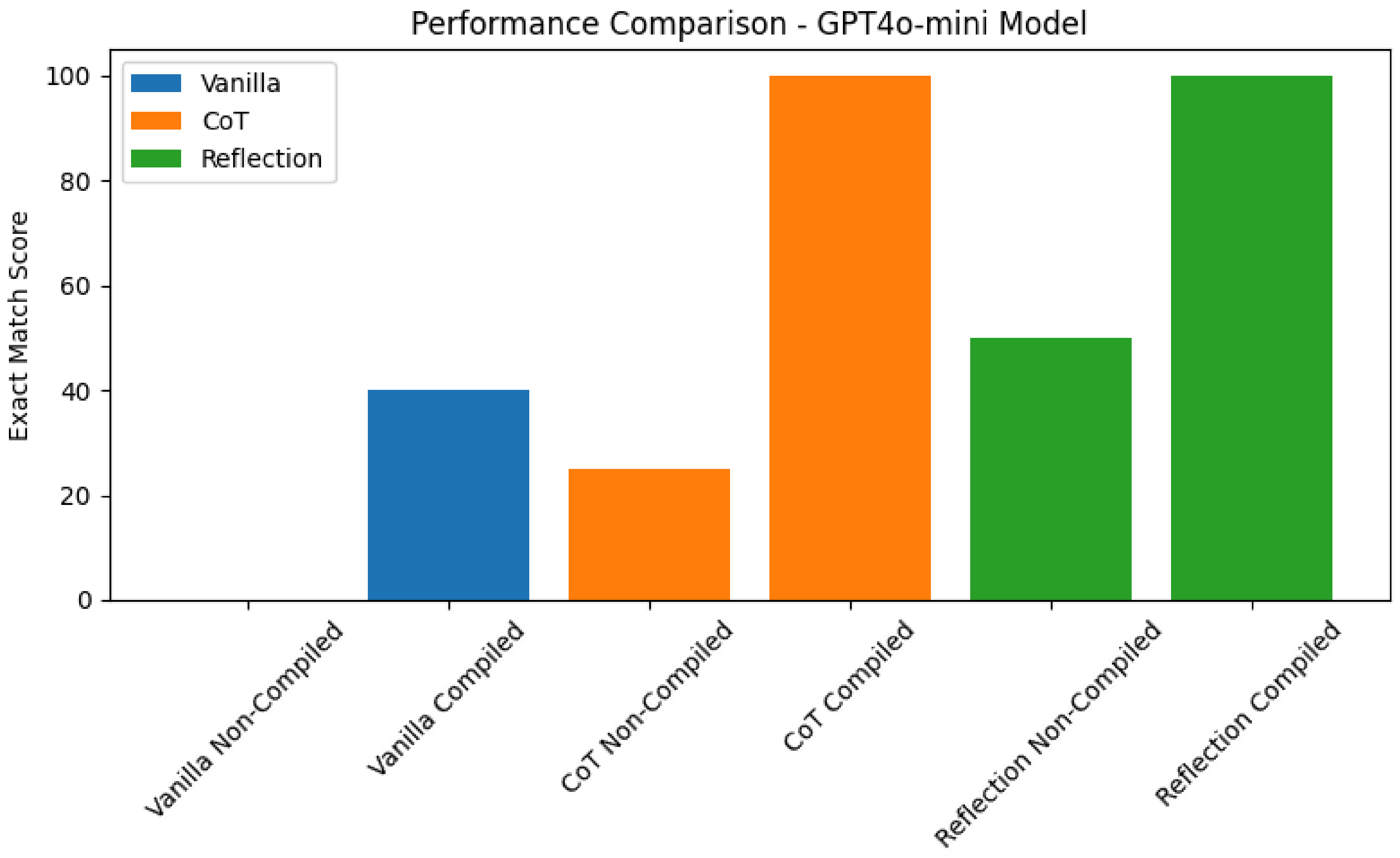
Reproduction de DSPy

- Comparaison entre les modules (Vanilla / CoT / Reflection)
- Comparaison entre avant/après optimisation

Table 2: Prediction Reasoning for Wendy's Dentist Bill Problem

Program	**Prediction Reasoning**
Vanilla	Wendy paid \$600 for the tooth extraction. *(incorrect)*
Vanilla + DSPy Compilation	Wendy paid \$240 for the tooth extraction. *(incorrect)*
CoT	Wendy paid \$150 for the tooth extraction. *(incorrect)*
CoT + DSPy Compilation	Let's denote the cost of the tooth extraction as x. Wendy's bill was five times the cost of a filling, which is \$120. So: \$70 (cleaning) + 2 × \$120 (fillings) + x = \$600. $\Rightarrow 70+240+x=600 \Rightarrow x=600-310=\boxed{290}$. *(correct)*
Reflection	The cost of the tooth extraction is \$250. *(incorrect)*
Reflection + DSPy Compilation	Given that Wendy's dentist bill was five times the cost of a filling, and each filling costs \$120, the total bill is 5 × \$120 = \$600. The bill includes the cleaning (\$70) and 2 fillings (\$120 each), so: \$70 + 2 × \$120 = \$310. Therefore, the cost of the tooth extraction is: \$600 - \$310 = $\boxed{290}$. *(correct)*

Extension sur GPT4o-mini



Comparaison GPT3.5 / GPT4o-mini

Program	GPT-3.5	GPT-4o-mini
Vanilla (Module Predict)	1.67	0.0
+ DSPy Compilation	38.33	40.0
CoT	8.00	20.0
+ DSPy Compilation	83.67	100.0
Reflection	12.67	35.0
+ DSPy Compilation	85.33	100.0

Table: Accuracy comparison of uncompiled and compiled DSPy modules across GPT-3.5 and GPT-4o-mini.

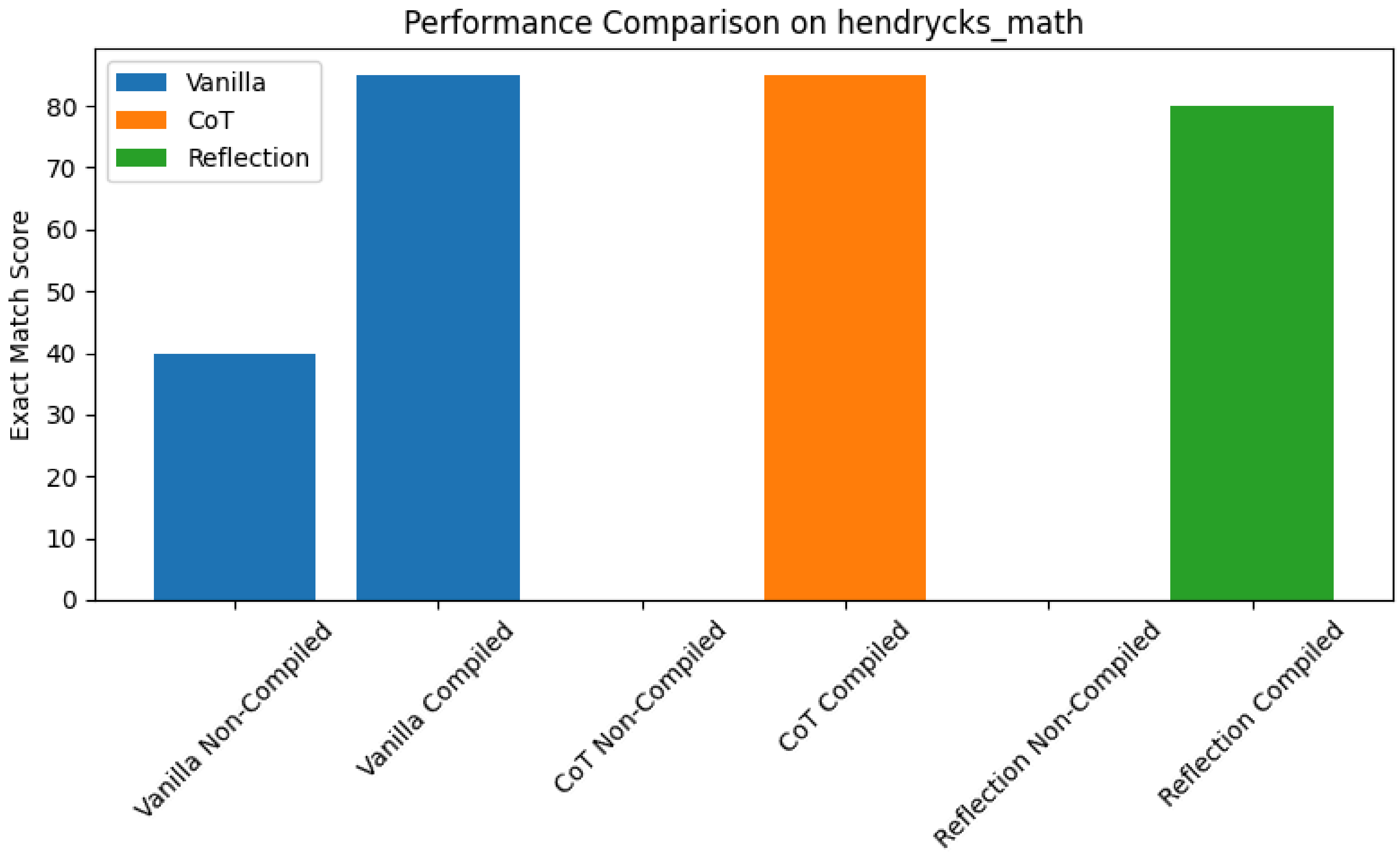
Comparaison GPT3.5 / GPT4o-mini

- Comparaison entre le raisonnement de chaque module

Table 3: Comparison of DSPy-Compiled Reasoning between GPT-3.5 and GPT-4o-mini

Model	Program (Compiled)	Prediction Reasoning
GPT-3.5	CoT + DSPy Compilation	Let's denote the cost of the tooth extraction as x. Wendy's bill was five times the cost of a filling, which is \$120. So: \$70 (cleaning) + 2 × \$120 (fillings) + $x = 600$. $\Rightarrow 70 + 240 + x = 600 \Rightarrow x = 600 - 310 = \boxed{290}$. *(correct)*
	Reflection + DSPy Compilation	Wendy's dentist bill was five times the cost of a filling. Each filling is \$120, so: \$600. Cleaning: \$70, Fillings: 2 × \$120 = \$240. Total known = \$310, Tooth extraction = \$600 - \$310 = $\boxed{290}$. *(correct)*
GPT-4o-mini	CoT + DSPy Compilation	The bill is 5 × \$120 = \$600. Cleaning = \$70, Fillings = 2 × \$120 = \$240. Total known = \$310, Tooth extraction = \$600 - \$310 = $\boxed{290}$. *(correct)*
	Reflection + DSPy Compilation	Charges: \$70 (cleaning), 2 × \$120 = \$240 (fillings). Cleaning + fillings = \$310. Total = 5 × \$120 = \$600. Tooth extraction = \$600 - \$310 = $\boxed{290}$. *(correct)*

Extension sur un nouveau dataset



Majority voting on a single model

- Définition du mini_bench (issue du GSM8k dataset)
- Initialisation de notre modèle “Qwen2.5-Coder-1.5B-Instruct”

Table 1: Sample from mini_bench (GSM8K Test Subset)

#	Question (abridged)	Answer
1	Janet's ducks lay 16 eggs/day. She eats 3 and uses 4 for muffins. She sells the rest at \$2 each. How much does she earn per day?	18
2	A robe takes 2 bolts of blue fiber and half that much white. How many total bolts?	3
3	Josh buys a house for \$80k, renovates for \$50k. The value increases 150%. What's his profit?	70000
4	James runs 3 sprints, 3×/week. Each sprint is 60m. How many meters per week?	540
5	Wendi feeds each chicken 3 cups/day. She gives 15 cups AM, 25 PM. Her flock has 20 chickens. How much for the last meal?	20
6	Kylar buys 16 glasses: 1st is \$5, 2nd is 60% of price. What's the total cost?	64
7	Toulouse = 2×Charleston; Charleston = 4×Seattle; Seattle has 20 sheep. How many total sheep?	260
8	Carla downloads 200GB. At 2GB/min, she restarts at 40% (after 20min pause). How long to redownload fully?	160
9	John drives 3h at 60mph. On return: 2h traffic (0 mph), 0.5h at 30mph, then 1.5h at 80mph. How far is he from home after 4h?	45
10	Eliza earns \$10/hr for 40h/week. Overtime = 1.2×rate. If she works 45h this week, how much does she earn?	460

Majority voting on a single model

- Implémentation du Majority voting avec N candidat(s)

Algorithm 1 GenerateCandidates

Input: Math question q, number of generations N
Output: List of N decoded responses $\mathcal{R}$

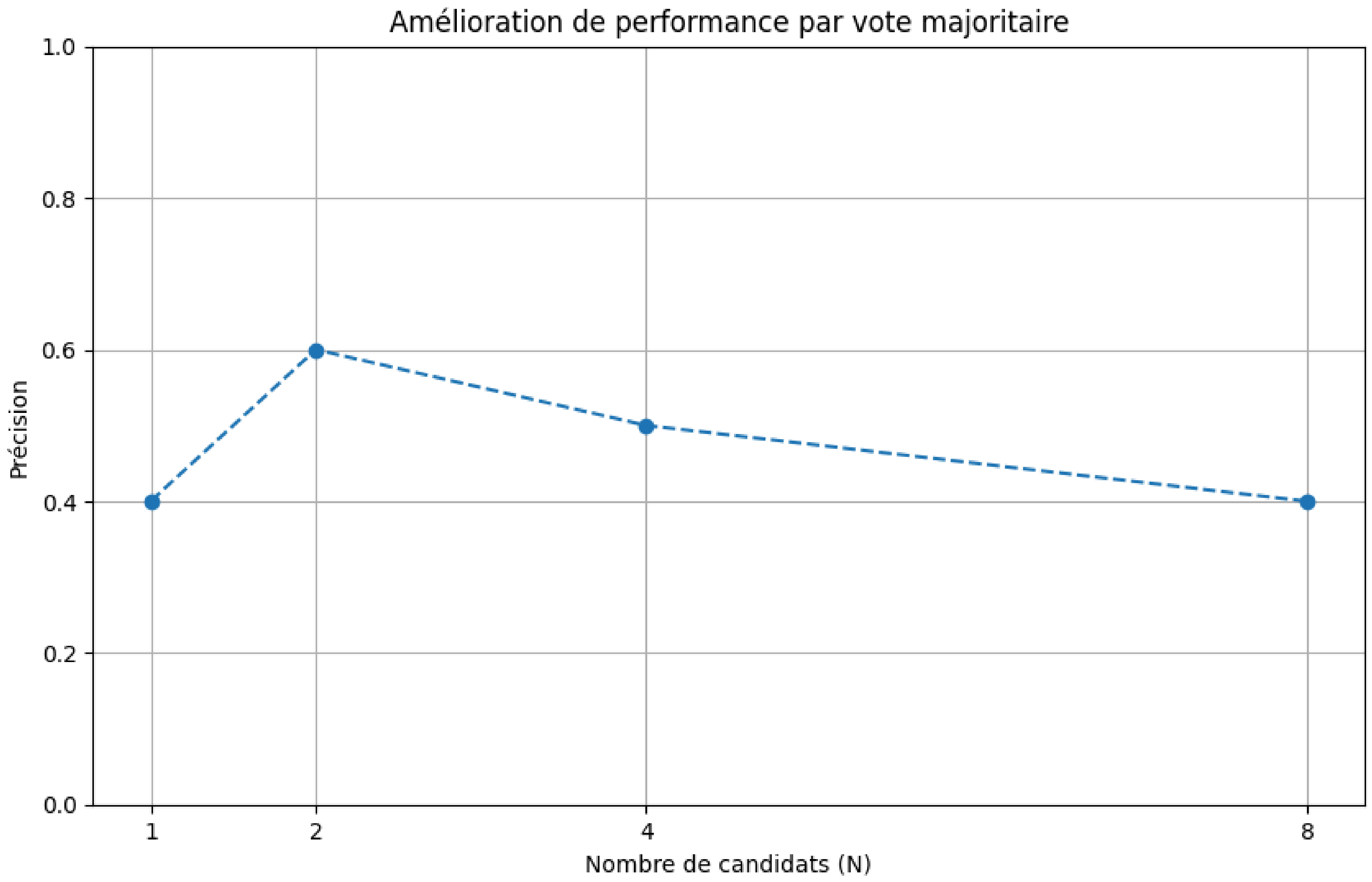
1: Construct a structured prompt P containing q.
2: Tokenize P and send to LLM using sampling-based decoding with:
   - `num_return_sequences` $= N$
   - `do_sample = True`, `temperature = 0.7`, `top_p = 0.9`
3: For each of the N outputs:
   - Decode the output into text.
   - Extract Python code between ‘‘‘`python` ... ‘‘‘.
4: **return** $\mathcal{R}$, the list of code blocks

Algorithm 2 MajorityVote

Input: List of numerical outputs $A = \{a_1, a_2, \ldots, a_n\}$
Output: Most frequent result a^*

1: Filter out any `None` or invalid results from A.
2: Count occurrences of each remaining value using a frequency map F.
3: Identify the maximum frequency $f^* = \max F[a_i]$.
4: Let $C = \{a_i \in A \mid F[a_i] = f^*\}$ be the tied candidates.
5: Break ties by selecting a random $a^* \in C$.
6: **return** a^*

Majority voting on a single model



Multi-Model Voting

- Implémentation du Majority voting cross-models

Algorithm 3 GenerateAnswer

Input : Math question q, model $\mathcal{M}$, tokenizer $\mathcal{T}$
Output: Normalized numerical result r

Format prompt P to instruct model to generate Python code
  Tokenize prompt: inputs $\leftarrow \mathcal{T}(P)$
  Run model generation: output $\leftarrow \mathcal{M}$(inputs)
  Extract code block delimited by ‘‘‘`python ...`‘‘‘
  if *code is valid* **then**
    Execute code using `exec()`
    Retrieve variable `result` from local scope
    Normalize numerical value using `float()`
    return r
else
  return *None*

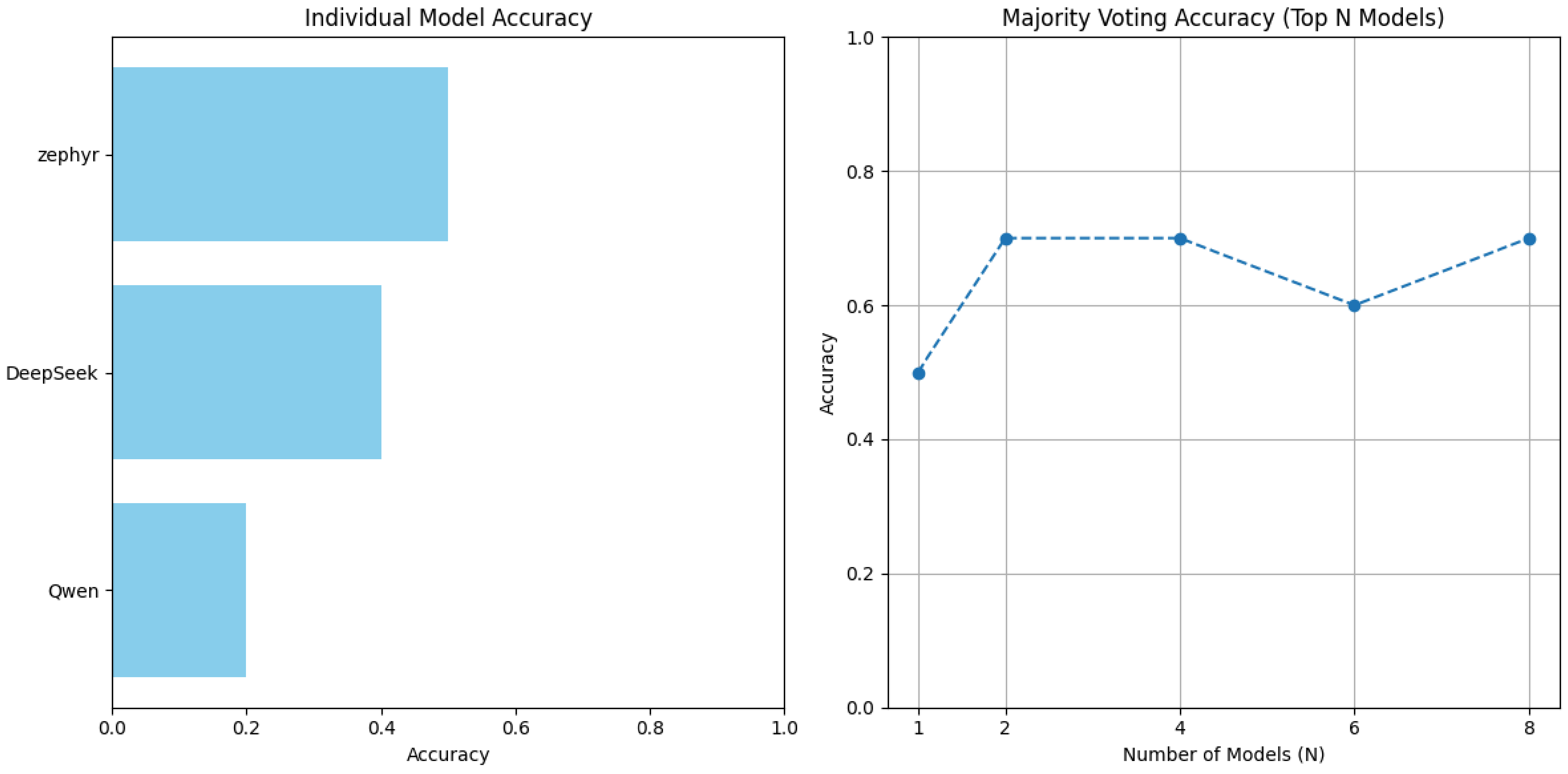
Algorithm 5 EvaluateModelsAndMajority

Benchmark set $\mathcal{B}$, model set $\mathcal{M}_1, \ldots, \mathcal{M}_k$ Individual accuracies and ensemble accuracies

foreach *model* $\mathcal{M}_i$ **do**
  foreach *question-answer pair* (q, a) *in* $\mathcal{B}$ **do**
    $r_i \leftarrow$ GenerateAnswer(q, $\mathcal{M}_i$)
    if $r_i == a$ **then**
      Increment correct count for $\mathcal{M}_i$
  Compute accuracy as $\frac{\text{correct}}{|\mathcal{B}|}$
Sort models by accuracy into list $\mathcal{M}_{sorted}$

foreach N *in top-N values* **do**
  foreach (q, a) *in* $\mathcal{B}$ **do**
    Let $R \leftarrow$ list of predictions $r_1, \ldots, r_N$ from top-N models
    $r^* \leftarrow$ MajorityVote(R)
    if $r^* == a$ **then**
      Increment ensemble correct count
  Compute ensemble accuracy as $\frac{\text{correct}}{|\mathcal{B}|}$

Multi-Model Voting



Conclusion

Contribution:

- Expérimentation et reproduction du papier DSPy
- Extension sur un nouveau modèle et nouveau dataset
- Imitation avec une approche intuitive: LLM Ensembling

Travaux futurs :

- Comparer DSPy et GRPO (cf TPs) : coût vs performance.
- Étendre à d’autres tâches complexes (Q&A).
- Tester les modules DSPy avec mise à jour de poids.

Références

[1] Omar Khattab, Ankit Singhvi, Pratyush Maheshwari, Zhun Zhang, Keshav Santhanam, Satyapriya Vardhamanan, Syed Haq, Ananya Sharma, Tanmayi Thyagarajan Joshi, Humza Moazam, Houston Miller, Matei Zaharia, Christopher Potts.
DSPy: Compiling Declarative Language Model Calls into Self-Improving Pipelines.
Stanford University, UC Berkeley, Amazon Alexa AI, CMU, Microsoft, IIT Bombay, Dashworks, Calera Capital, Two Sigma Investments.
📄 Link to paper – arXiv:2402.19150, 2024.

[2] Omar Khattab, Keshav Santhanam, Xiang Lisa Li, David Hall, Percy Liang, Christopher Potts, Matei Zaharia.
Demonstrate-Search-Predict: Composing Retrieval and Language Models for Knowledge-Intensive NLP.
📄 arXiv:2212.14024, 2022.

[3] Jinyi Xiang, Jing Zhang, Zhengyu Yu, Fengwei Teng, Jialiang Tu, Xiaodan Liang, Shilin Hong, Changzhou Wu, Yiyu Luo.
Self-Supervised Prompt Optimization.
📄 arXiv link, 2023.

[4] Ziyang Shao, Peng Wang, Qian Zhu, Runxin Xu, Jiaxin Song, Xiaodong Bi, Hang Zhang, Meng Zhang, Yikang Li, Yuxuan Wu, Dongyan Guo.
DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models.
DeepSeek-AI, Tsinghua University, Peking University.
🔗 GitHub Repository

[5] OpenAI API Key
Utilisé pour tester les modèles avancés (comme GPT-4o-mini) via appel API dans les modules DSPy.

[6] LLM MVA Course – ENS Paris-Saclay
Supports de cours, TP1 & TP2 disponibles sur : 🔗 https://llm.labri.fr/#slides

[7] Qwen/Qwen2.5-Coder-1.5B-Instruct
Modèle open-source Hugging Face : 🔗 https://huggingface.co/Qwen/Qwen2.5-Coder-1.5B-Instruct

Merci !