

Лекция 10: Advanced Prompting и Reasoning Models

От прости prompts до модели, които "мислят"

Цели на лекцията

- Проблемът с elicitation — как да извлечем възможностите
- Chain-of-Thought — революционната техника
- Few-shot vs Zero-shot prompting
- Self-consistency и Tree of Thoughts
- Reasoning models — o1, DeepSeek R1
- Практически framework за избор на подход

Част 1: Проблемът с Elicitation

Същият модел, различни резултати

Въпрос: Roger има 5 топки за тенис. Купува още 2 кутии с по 3 топки. Колко топки има?

Директен prompt:

Model: 11

(понякога грешен)

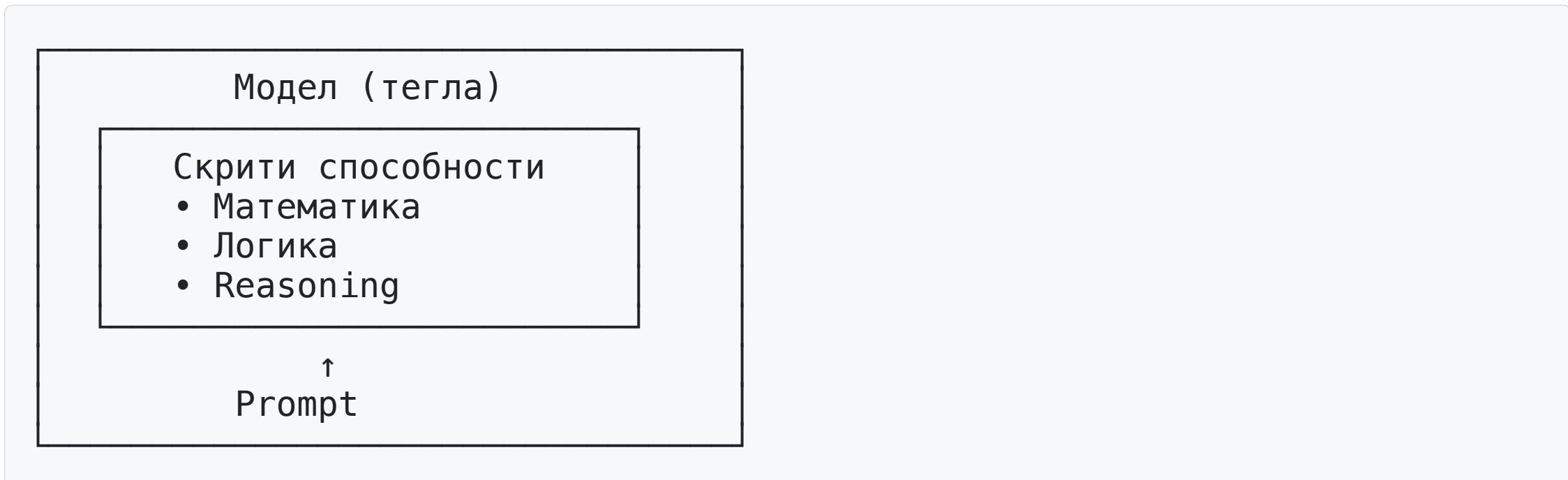
С "Let's think step by step":

Model: Roger има 5 топки. Купува 2 кутии \times 3 топки = 6.
Общо: $5 + 6 = 11$ топки.

(почти винаги верен)

Защо има разлика?

- Моделът **знае** как да реши задачата
- Но не винаги го **прави** spontaneously
- Prompting = "извикване" на скрити способности



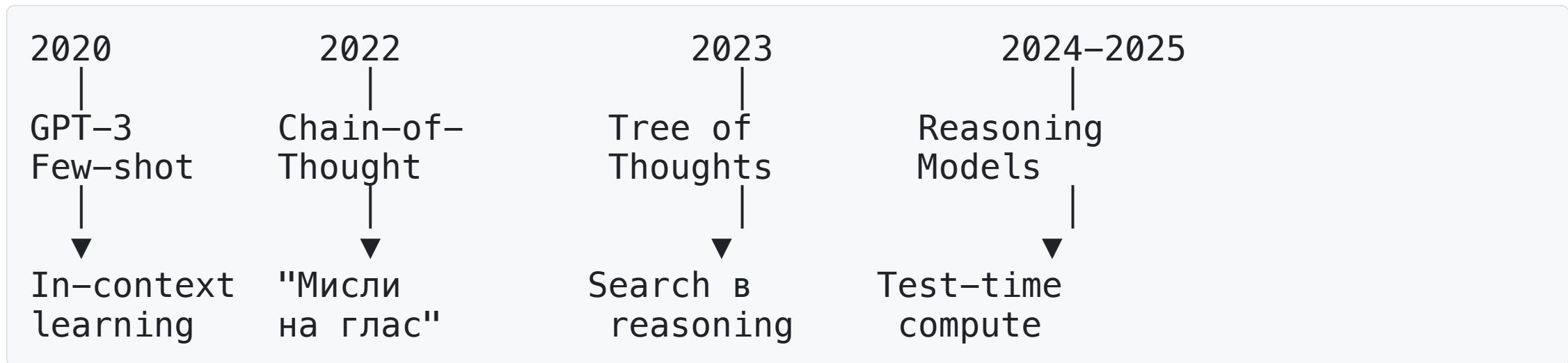
Prompting като "програмиране"

Традиционно	Prompting
Промяна на код	Промяна на входа
Compile/Train	Inference
Часове/дни	Секунди
Нужна експертиза	Експериментиране

Ключов insight: Не променяме теглата — променяме как модела ги използва

Част 2: Еволюция на Prompting техниките

Timeline



2020: Few-Shot Learning (GPT-3)

Brown et al. — "Language Models are Few-Shot Learners"

Преведи English → French:

sea otter → loutre de mer

cheese → fromage

hello → bonjour

cat →

Открытие: Примери в prompt-а → модел "научава" pattern

Шок: Без fine-tuning за много задачи!

2022: Chain-of-Thought

Wei et al. — "Chain-of-Thought Prompting Elicits Reasoning"

Ключов insight: Показването на стъпки подобрява точността

Q: Магазин продава ябълки по 2 лв и круши по 3 лв.
Ако купя 4 ябълки и 2 круши, колко плащам?

A: Нека помислим стъпка по стъпка.

$$\text{Ябълки: } 4 \times 2 \text{ лв} = 8 \text{ лв}$$

$$\text{Круши: } 2 \times 3 \text{ лв} = 6 \text{ лв}$$

$$\text{Общо: } 8 + 6 = 14 \text{ лв}$$

Zero-Shot CoT

Kojima et al. — "Large Language Models are Zero-Shot Reasoners"

Магическа фраза: "Let's think step by step"

Before: "Колко е 17×24 ?" → "408" ✗

After: "Колко е 17×24 ? Let's think step by step."
→ " $17 \times 24 = 17 \times 20 + 17 \times 4$
 $= 340 + 68 = 408$ " ✓

Без примери. Само фразата. Работи.

2022-2023: Refinements

Self-Consistency (Wang et al.)

- Генерирай N reasoning chains
- Вземи majority vote
- +5-15% точност

Tree of Thoughts (Yao et al.)

- Search през reasoning paths
- Backtrack когато е нужно

2024-2025: Reasoning Models

Нов paradigm: Моделите са обучени да reasoning-ват

- OpenAI o1/o3: Test-time compute scaling
- DeepSeek R1: Open weights, visible reasoning
- Claude Extended Thinking: Configurable depth

По-дълго мислене = по-добри резултати

Част 3: Prompt Engineering Fundamentals

Принцип 1: Яснота и специфичност

Лошо:

Напиши за Python.

Добро:

Напиши функция на Python, която приема списък от числа и връща сумата на четните числа.
Включи docstring и примери за използване.

Принцип 2: Output формат

Без формат:

Анализирай sentiment на този текст: "Храната беше ужасна"
→ Текстът изразява негативно мнение за храната...

С формат:

Анализирай sentiment. Отговори само с: POSITIVE/NEGATIVE/NEUTRAL
"Храната беше ужасна"
→ NEGATIVE

Принцип 3: Структурни техники

Delimiters:

Summarize the following text:

```
###  
[Дълъг текст тук]  
###
```

XML tags:

```
<context>  
Текст за анализ  
</context>
```

```
<instruction>  
Извлечи ключовите точки  
</instruction>
```

Принцип 4: Позиция на инструкции

Option A:

Instructions
Context
...

Option B:

Context
...
Instructions



По-добро за дълъг context

Tip: Инструкции в края работят по-добре при дълъг контекст

Честа грешка: "Don't"

Проблем:

Don't mention that you're an AI.

→ Модел вероятно ще спомене

По-добре:

Respond as a helpful assistant focused only on the topic.

Зашо: Моделът генерира въз основа на prompt-а — "AI" е в контекста

Част 4: Chain-of-Thought Deep Dive

Защо СоT работи?

LLM-ите нямат "скрито изчисление":

Predict token → Predict token

Няма "мислене" между токени

СоТ форсира "мислене" в текста

Intermediate steps = working memory

Zero-Shot CoT в действие

Тип задача	Без CoT	С CoT	Подобреие
Math Word Problems	17.7%	78.7%	+61%
Multi-Step Logic	52%	83%	+31%
Common Sense	73%	75%	+2%

Извод: CoT помога най-много на reasoning задачи

Few-Shot CoT

Q: Джон има 3 пъти повече марки от Мария.
Мария има 4 марки. Колко има Джон?

A: Мария има 4 марки.
Джон има $3 \times 4 = 12$ марки.
Отговор: 12

Q: В клас има 24 ученици. 1/3 са момичета.
Колко са момчетата?

A: [Модел продължава pattern-а]

Кога да използваме СоТ?

Помага при:

- Математика с повече стъпки
- Логически пъзели
- Задачи със заблуждаващи условия
- Code debugging

Не помага при:

- Прости factual въпроси
- Translation
- Text summarization
- Creative writing

Self-Consistency

Problem → Generate 5 CoT chains ($\text{temp} > 0$)

Chain 1: ... Answer: 42

Chain 2: ... Answer: 38

Chain 3: ... Answer: 42

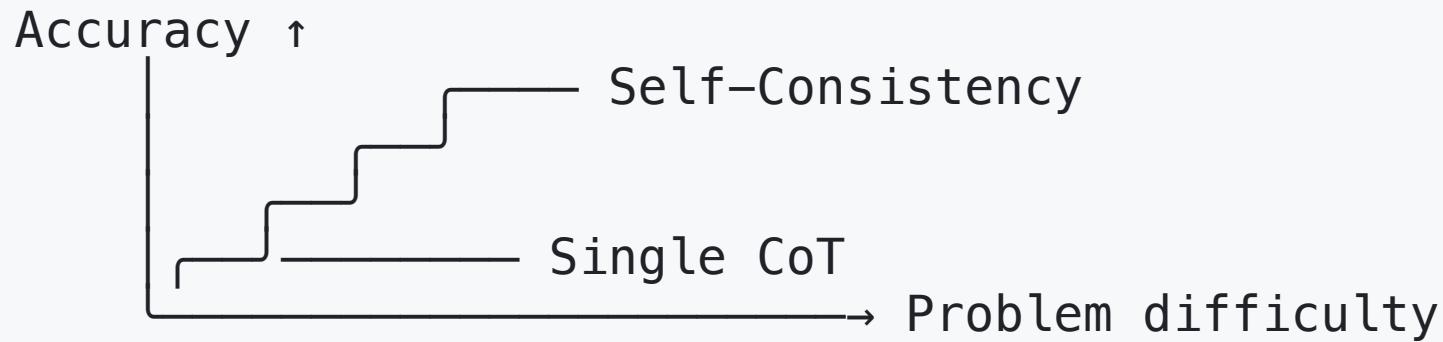
Chain 4: ... Answer: 42

Chain 5: ... Answer: 40

Majority vote → 42

Trade-off: 5× cost за +5-15% accuracy

Self-Consistency: Кога?



По-голяма полза при по-трудни задачи

Част 5: Tree of Thoughts

Ограничение на линейния СоТ

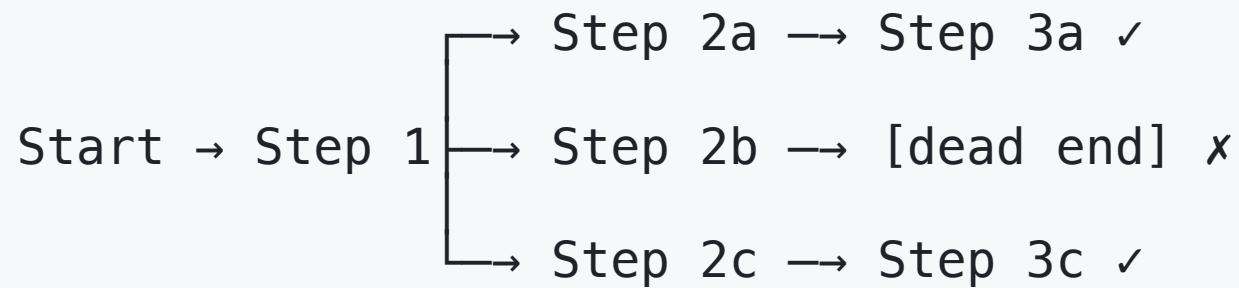
Step 1 → Step 2 → Step 3 → Answer

└ Ако грешен, цялата верига е компрометирана

Проблем: Една грешка = грешен отговор

Няма: Backtracking, алтернативи, exploration

ToT: Идеята



- Генерирай **множество** следващи стъпки
- Оцени кои са promising
- Explore добрите, prune лошите

Classic Example: Game of 24

Задача: С 4 числа (напр. 1, 2, 3, 4) и +, -, ×, / направи 24

Числа: 4, 7, 8, 8

ToT exploration:

- 8 / 4 = 2, остават 2, 7, 8 → [dead end]
- 8 - 4 = 4, остават 4, 7, 8
 - (8 - 4) × (7 - 1)... hmm, няма 1
- 7 + 8 = 15, остават 4, 8, 15
 - ...
- 8 × (7 - 4) = 24! ← но трябва и 8...
 - 8 × 3 = 24, нека видим: $(8 - (8/8)) \times 4$? Не.
 - Продължава...

ToT: Практическа реалност

Предимства:

- По-добра точност на hard problems
- Може да открие грешки и backtrack

Недостатъци:

- Много скъпо (много LLM calls)
- Бавно
- Рядко в production

Важно концептуално: Reasoning as search → предвестник на reasoning models

Част 6: Reasoning Models

Paradigm Shift

Старият подход:

User → Prompt Engineering → Model → Answer
(човек оптимизира)

Новият подход:

User → Reasoning Model → Extended Thinking → Answer
(модел сам мисли по-дълго)

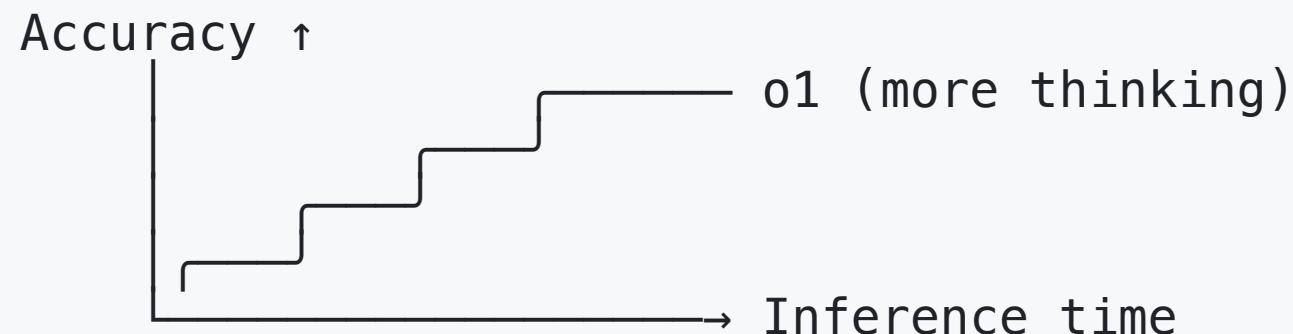
Test-Time Compute Scaling

Традиционно:

More training compute → Better model
(fixed inference cost)

Reasoning models:

More inference compute → Better answers
(model "thinks longer")



Process vs Outcome Supervision

Outcome supervision (стандартен RLHF):

Problem → ... → Final Answer → ✓/✗ Reward
(само тук)

Process supervision:

Problem → Step1 → Step2 → Step3 → Answer
✓ ✓ ✓ ✓
(reward на всяка стъпка)

Предимство: Хваща "lucky guesses" с грешен reasoning

Как работят Reasoning Models

Extended Thinking

Try approach A... evaluate... не работи
Try approach B... evaluate... promising
Verify step by step...
Double-check calculation...



Final Answer

Текущият Landscape

Model	Reasoning	Open	Visible Thinking
o1/o3	Strong	✗	Hidden
DeepSeek R1	Strong	✓	Visible
Claude	Configurable	✗	Optional

DeepSeek R1: Пример

Question: Колко дни в седмица съдържат буквата "а"?

<think>

Нека изброя дните: понеделник, вторник, сряда,
четвъртък, петък, събота, неделя.

Понеделник – има 'а' ✓

Вторник – няма 'а' ✗

Сряда – има 'а' ✓

Четвъртък – няма 'а' ✗

Петък – няма 'а' ✗

Събота – има 'а' ✓

Неделя – има 'а' ✓

</think>

Четири дни: понеделник, сряда, събота, неделя.

Кога Reasoning Model?

Подходящо:

- Сложна математика
- Code generation/debugging
- Multi-step logic
- High-stakes decisions

Неподходящо:

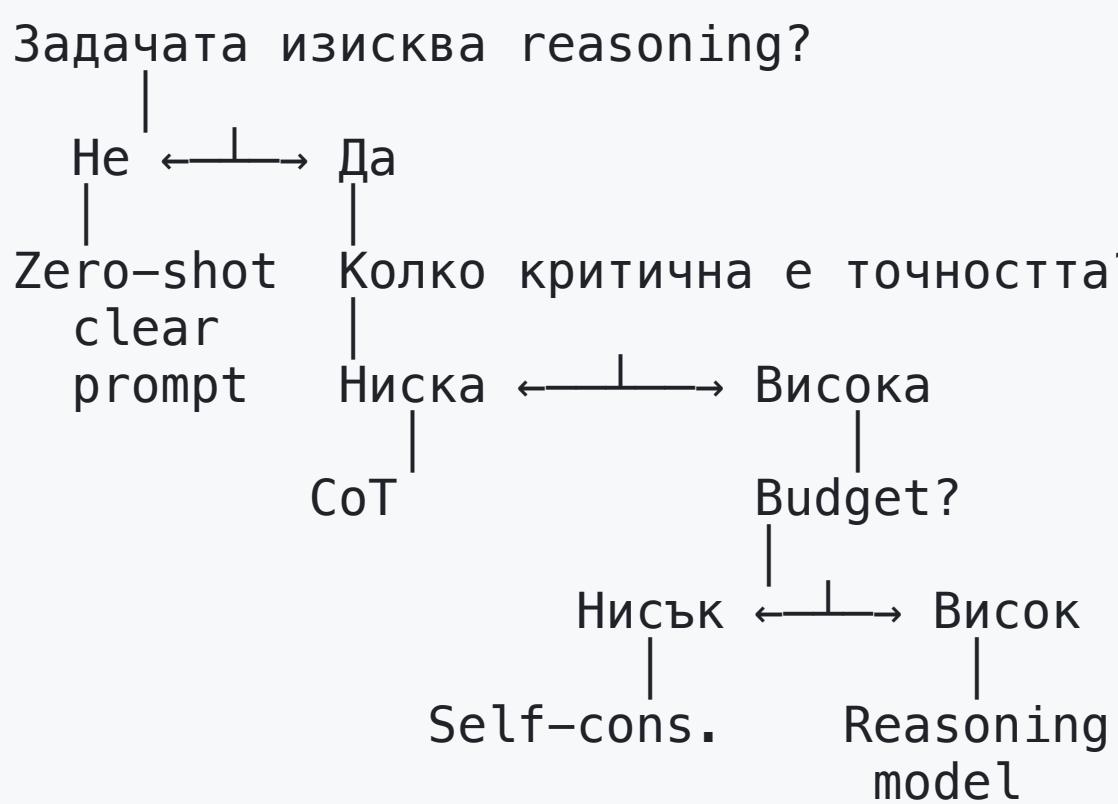
- Прости въпроси (overkill)
- Creative writing
- Conversation/chat
- Когато latency е критична

Cost-Benefit Analysis

Подход	Latency	Cost	Best For
Standard model	~1s	1×	Повечето задачи
Standard + CoT	~2-3s	1×	Reasoning on budget
Self-consistency	~5-10s	5×	Когато accuracy > cost
Reasoning model	10s-5min	5-20×	Critical reasoning

Част 7: Практически Framework

Decision Tree



Стъпка 1: Оцени задачата

Прости задачи:

- Factual Q&A
- Format conversion
- Translation
- Simple classification

Reasoning задачи:

- Math word problems
- Multi-step logic
- Code debugging
- Complex analysis

Стъпка 2: Започни просто

1. Ясен, специфичен prompt



Не работи?



2. Добави format instructions



Не работи?



3. Добави СоT ("Let's think step by step")



Не работи?



4. Few-shot examples с reasoning



Критична точност?



5. Self-consistency или Reasoning model

Стъпка 3: Measure

```
# Не гадай – измервай!
prompts = [
    "Simple: {question}",
    "CoT: {question} Let's think step by step.",
    "Few-shot: [examples] {question}",
]

for prompt in prompts:
    accuracy = evaluate(prompt, test_set)
    print(f"{prompt}: {accuracy:.2%}")
```

Практически Tips

1. СоT е бесплатно подобрение за reasoning задачи

- Винаги пробвай преди да усложняваш

2. Few-shot качеството > quantity

- 3 добри примера > 10 лоши

3. Consistency има diminishing returns

- След 5-7 samples подобрението намалява

4. Reasoning models са за специални случаи

- Не ги ползвай за всичко

Обобщение

Ключови идеи

1. **Prompting** е "програмиране" без промяна на тегла
2. **Еволюция:** Few-shot (2020) → CoT (2022) → Reasoning models (2024)
3. **CoT:** "Let's think step by step" — просто и ефективно
4. **Self-consistency:** Majority voting за повече точност
5. **Reasoning models:** Test-time compute scaling, process supervision
6. **Framework:** Започни просто, добавяй сложност при нужда

Следваща лекция

Лекция 11: Hallucinations и RAG

- Защо моделите халюцинират
- Embeddings и semantic search
- Vector databases
- Retrieval-Augmented Generation

Ресурси

Papers:

- Wei et al. (2022) — Chain-of-Thought
- Kojima et al. (2022) — Zero-shot Reasoners
- Wang et al. (2022) — Self-Consistency
- Lightman et al. (2023) — Let's Verify Step by Step

Survey:

- Schulhoff et al. (2024) — The Prompt Report

Въпроси?