AlphaCode

Seminarium: Reinforcement learning dla gier

Jakub Kuciński

Opis problemu w competitive programming

M. Mirzayanov. Codeforces: Results of 2020

Backspace

You are given two strings s and t, both consisting of lowercase English letters. You are going to type the string s character by character, from the first character to the last one.

When typing a character, instead of pressing the button corresponding to it, you can press the "Backspace" button. It deletes the last character you have typed among those that aren't deleted yet (or does nothing if there are no characters in the current string). For example, if s is "abcbd" and you press Backspace instead of typing the first and the fourth characters, you will get the string "bd" (the first press of Backspace deletes no character, and the second press deletes the character 'c'). Another example, if s is "abcaa" and you press Backspace instead of the last two letters, then the resulting text is "a".

Your task is to determine whether you can obtain the string t, if you type the string s and press "Backspace" instead of typing several (maybe zero) characters of s.

Input

The first line contains a single integer q ($1 \le q \le 10^5$) the number of test cases.
The first line of each test case contains the string s ($1 \le |s| \le 10^5$). Each character of s is a lowercase English letter.
The second line of each test case contains the string t ($1 \le |t| \le 10^5$). Each character of t is a lowercase English letter.
It is guaranteed that the total number of characters in the strings over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, print "YES" if you can obtain the string t by typing the string s and replacing some characters with presses of "Backspace" button, or "NO" if you cannot.
You may print each letter in any case (YES, yes, Yes will all be recognized as positive answer, NO, no and nO will all be recognized as negative answer).

Example Input

```
4
ababa
ba
ababa
bb
aaa
aaaa
aababa
ababa
```

Example Output

```
YES
NO
NO
YES
```

Explanation

In order to obtain "ba" from "ababa", you may press Backspace instead of typing the first and the fourth characters.

There's no way to obtain "bb" while typing "ababa".

There's no way to obtain "aaaa" while typing "aaa".

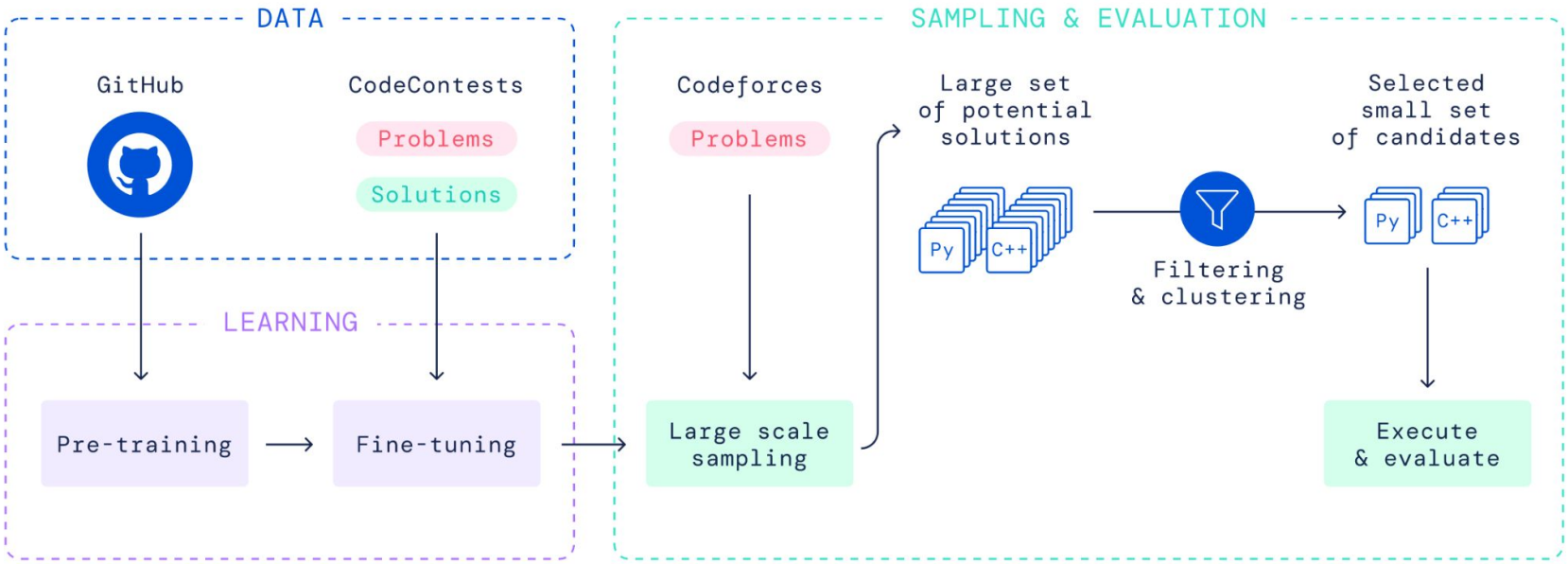
In order to obtain "ababa" while typing "aababa", you have to press Backspace instead of typing the first character, then type all the remaining characters.

```python
t=int(input())
for i in range(t):
    s=input()
    t=input()
    a=[]
    b=[]
    for j in s:
        a.append(j)
    for j in t:
        b.append(j)
    a.reverse()
    b.reverse()
    c=[]
    while len(b)!=0 and len(a)!=0:
        if a[0]==b[0]:
            c.append(b.pop(0))
            a.pop(0)
        elif a[0]!=b[0] and len(a)!=1:
            a.pop(0)
            a.pop(0)
        elif a[0]!=b[0] and len(a)==1:
            a.pop(0)
    if len(b)==0:
        print("YES")
    else:
        print("NO")
```

Wynik systemu

Yujia Li et al. (2022), Competition-Level Code Generation with AlphaCode

AlphaCode


DATA
GitHub
CodeContests
Problems
Solutions
LEARNING
Pre-training
Fine-tuning
SAMPLING & EVALUATION
Codeforces
Problems
Large scale sampling
Large set of potential solutions
Py
C++
Filtering & clustering
Selected small set of candidates
Py
C++
Execute & evaluate

Datasety

Pre-training

Fine-tuning

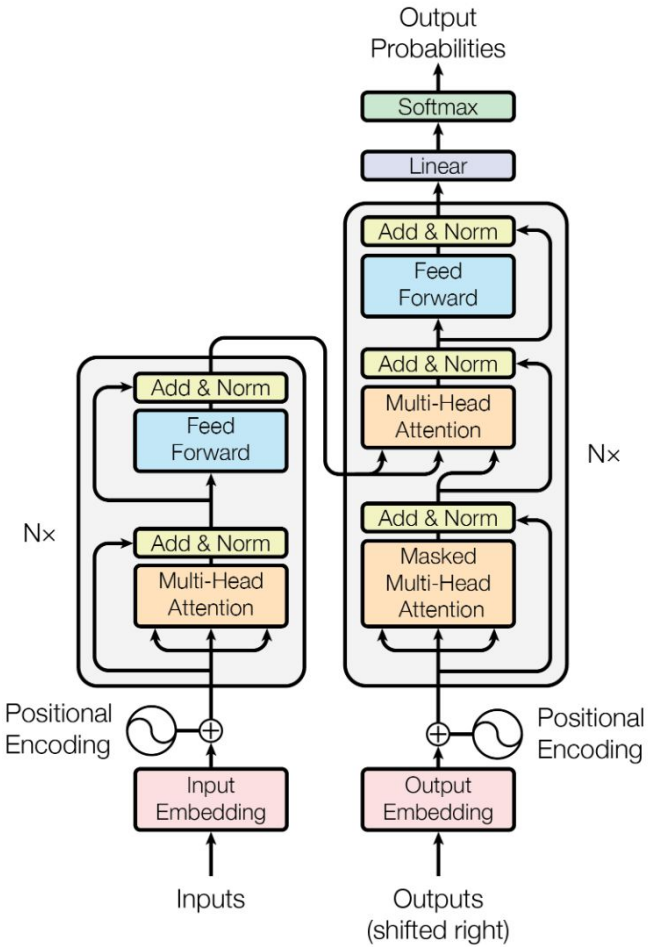
CODEFORCES

CodeNet

Description2Code

Encoder-decoder transformer

- Płytki, ale dłuższy encoder
- Głęboki, ale krótszy decoder
- Wspólny tokenizator
- Współdzielone macierze K i V pomiędzy headami dla szybszego samplowania
- Cross-entropy na predykcji kolejnego tokenu dla decodera i Masked Language Modeling Loss dla encodera



Ashish Vaswani et al. (2017), Attention Is All You Need

Fine-tuning

Tempering

Przed softmaxem dzielimy liczby przez T<1, co wyostrza rozkład, przez co sieć uczy się zwracać gładsze rozkłady.

Metadata conditioning

Podczas fine-tuningu dodajemy metadane, podczas inferencji możemy nimi manipulować.

GOLD

Mamy wiele rozwiązań do każdego problemu. Nie chcemy średniej z nich, tylko dowolne poprawne.

```
RATING: 1200
TAGS: dp,implementation
LANGUAGE IS python3
CORRECT SOLUTION
Polycarp must pay exactly n burles at the checkout ... (rest of the description)
```

GOLD - motywacja

Problemy MLE:

- Inna funkcja celu w czasie treningu niż metryka w czasie ewaluacji.
- Rozkłada prawdopodobieństwo po danych treningowych, co prowadzi do zróżnicowanych, ale też często słabych jakościowo predykcji.
- Exposure bias problem – uczymy model na złotym przedrostku, a podczas ewaluacji sekwencyjnie generujemy tokeny na podstawie poprzednich.

Richard Yuanzhe Pang and He He (2020). Text Generation by Learning from Off-Policy Demonstrations

MLE - trening i ewaluacja

- Kontekst $\boldsymbol{x}$, chcemy wygenerować $\boldsymbol{y} = (y_0, \ldots, y_T)$
- Modelujemy problem przez rozkład warunkowy:

$$p_\theta(\boldsymbol{y} \mid \boldsymbol{x}) = \prod_{t=0}^{T} p_\theta(y_t \mid \boldsymbol{y}_{0:t-1}, \boldsymbol{x})$$

- Rozkład z którego pochodzą dane: $p_{\text{human}}(\boldsymbol{y} \mid \boldsymbol{x})$
- Funkcja straty podczas treningu:

$$\mathcal{L}(\theta) = -\mathbb{E}_{\boldsymbol{y} \sim p_{\text{human}}} \left[\sum_{t=0}^{T} \log p_\theta(y_t \mid \boldsymbol{y}_{0:t-1}, \boldsymbol{x}) \right]$$

- Ewaluacja:

$$\mathbb{E}_{\boldsymbol{y} \sim p_\theta} \left[\sum_{t=0}^{T} \log p_{\text{human}}(y_t \mid \boldsymbol{y}_{0:t-1}, \boldsymbol{x}) \right]$$

Generowanie jako RL

- W kroku t polityka π_θ wybiera akcję a_t (odpowiadającą konkretnemu tokenowi), następuje przejście do stanu s_{t+1} i otrzymujemy nagrodę r_t.
- $s_{t+1} = (y_{0:t}, x)$
- $\pi_\theta(a_t|s_t) = p_\theta(a_t|y_{0:t}, x)$
- $r_t = p_{\text{human}}(a_t|s_t)$
- Cała sekwencja y może być wtedy zapisana jako trajektoria $\tau = (s_0, a_0, r_0, \ldots, s_T, a_T, r_T)$ (trajektorie z danych nazywamy demonstracjami).
- Chcemy maksymalizować oczekiwany zysk

$$J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{T} \gamma^t r_t \right]$$

Policy gradient

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_t \nabla_\theta \log \pi_\theta(a_t \mid s_t) \hat{Q}(s_t, a_t) \right]$$

$$\hat{Q}(s_t, a_t) = \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'}$$

Offline learning

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_t \nabla_\theta \log \pi_\theta(a_t \mid s_t) \hat{Q}(s_t, a_t) \right]$$

$$\mathbb{E}_{\tau \sim \pi_b} \left[\sum_t w_t \nabla_\theta \log \pi_\theta(a_t \mid s_t) \hat{Q}(s_t, a_t) \right]$$

$$w_t = \prod_{t'=0}^{t} \frac{\pi_\theta(a_{t'} | s_{t'})}{\pi_b(a_{t'} | s_{t'})}$$

Offline learning

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_t \nabla_\theta \log \pi_\theta(a_t \mid s_t) \hat{Q}(s_t, a_t) \right]$$

$$\mathbb{E}_{\tau \sim \pi_b} \left[\sum_t w_t \nabla_\theta \log \pi_\theta(a_t \mid s_t) \hat{Q}(s_t, a_t) \right]$$

$$w_t \approx \frac{\pi_\theta(a_t|s_t)}{\pi_b(a_t|s_t)}$$

Offline learning

$$\mathbb{E}_{\tau \sim \pi_b} \left[\sum_t w_t \nabla_\theta \log \pi_\theta(a_t \mid s_t) \hat{Q}(s_t, a_t) \right]$$

$$w_t \approx \frac{\pi_\theta(a_t|s_t)}{\pi_b(a_t|s_t)}$$

$\pi_b(\tau) \approx 1/N$ dla τ ze zbioru demonstracji i 0 w.p.p.

$$\nabla_\theta J(\theta) \approx \sum_{i=1}^{N} \sum_{t=0}^{T} \pi_\theta(a_t^i \mid s_t^i) \nabla_\theta \log \pi_\theta(a_t^i \mid s_t^i) \hat{Q}(s_t^i, a_t^i)$$

Off-policy policy gradient vs MLE gradient

$$\nabla_\theta J(\theta) \approx \sum_{i=1}^{N} \sum_{t=0}^{T} \pi_\theta(a_t^i \mid s_t^i) \nabla_\theta \log \pi_\theta(a_t^i \mid s_t^i) \hat{Q}(s_t^i, a_t^i)$$

$$\sum_{i=1}^{N} \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t^i \mid s_t^i)$$

δ-reward

$$r_t = R(s_t, a_t) \approx p_{\text{human}}(a|s)$$

$$R_\delta(s_t, a_t) \overset{\text{def}}{=} \begin{cases} 1, & \text{if } t = T \text{ and } (s_{0:T}, a_{0:T}) \in \mathcal{D} \\ 0, & \text{otherwise} \end{cases}$$

Estymowane p_{human}

$q(a|s)$ ma przybliżać $p_{\text{human}}(a|s)$

$$KL(p_{\text{human}} \| q) = E_{p_{\text{human}}}[\log p_{\text{human}}] - E_{p_{\text{human}}}[\log q]$$

Estymujemy p_{human} przy pomocy p_{MLE}

Nagrody

$$R_p(s,a) \stackrel{\text{def}}{=} \log p_{\text{MLE}}(a \mid s)$$
$$\hat{Q}_t(s_t,a_t) = \sum_{t'=t}^{T} \log p_{\text{MLE}}(a_t \mid s_t)$$

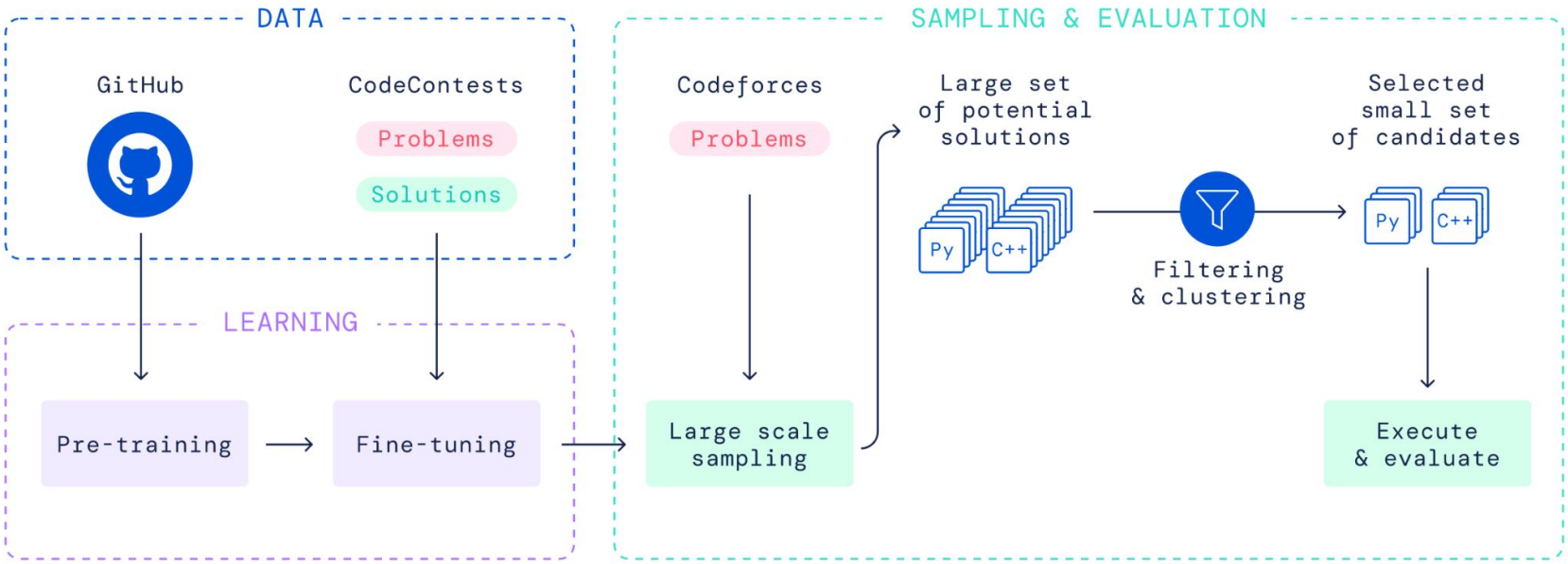
$$R_s(s,a) \stackrel{\text{def}}{=} p_{\text{MLE}}(a \mid s)$$
$$\hat{Q}(s_t,a_t) = \sum_{t'=t}^{T} p_{\text{MLE}}(a_t \mid s_t)$$

Generation by off-policy learning from demonstrations

Algorithm 1: GOLD

$\pi_\theta \leftarrow p_{\text{MLE}}$, $\tilde{\pi}_\theta \leftarrow p_{\text{MLE}}$
for $step = 1, 2, \ldots, M$ **do**
  Sample a minibatch $B = \{(\boldsymbol{x}^i, \boldsymbol{y}^i)\}_{i=1}^{|B|}$
  foreach (s_t^i, a_t^i) **do**
    Compute importance weights $\max(u, \tilde{\pi}_\theta)$, and compute returns $\hat{Q}(s_t^i, a_t^i) - b$
  Update θ by (4) using gradient descent
  if $step \,\%\, k = 0$ **then** $\tilde{\pi}_\theta \leftarrow \pi_\theta$
Return: π_θ

AlphaCode


DATA
GitHub
CodeContests
Problems
Solutions
LEARNING
Pre-training
Fine-tuning
SAMPLING & EVALUATION
Codeforces
Problems
Large scale sampling
Large set of potential solutions
Py
C++
Filtering & clustering
Selected small set of candidates
Py
C++
Execute & evaluate

Sampling i ewaluacja

Sampling

- Połowa sampli w C++ i połowa w pythonie
- W metadanych losowo wybierane tagi i ratingi, zawsze podane “CORRECT SOLUTION”
- Wyostrzanie rozkładów przez dzielenie przez temperaturę T’

Filtrowanie

- Odrzucanie rozwiązań, które nie przechodzą przykładowych testów
- Odrzuconych zostaje około 99% rozwiązań

Klasteryzacja

- Pomocnicza sieć neuronowa wyuczona do generowania dodatkowych inputów (niekoniecznie poprawnych)
- Łączenie w klastry programów, które zachowują się tak samo na dodatkowych inputach

Dziękuję za uwagę

Źródła

- Yujia Li et al. (2022), Competition-Level Code Generation with AlphaCode
- Richard Yuanzhe Pang and He He (2020). Text Generation by Learning from Off-Policy Demonstrations
- AlphaCode blog post
- Wizualizacja atencji w AlphaCode