Programmēšana: Difūzie tokens

Problēmas apraksts

Jānim apnika maksāt par ChatGPT abonementu, un viņš vēlas izstrādāt savu lielo valodas modeli (Large Language Model, LLM), lai palīdzētu sev un izstrādātājiem Lemon AI.

> ***LLM*** *ir neironu tīkli, kas ģenerē tekstu. Tie kodē valodu kā* ***tokenus*** *— teksta gabalus, piemēram, vārdus, vārdu daļas vai rakstzīmes (piemēram,* `Hello`, `AI`, `olymp`, `iad`, `!` *ir tokens, kas kodē* `Hello AI olympiad!`*).*

Tradicionālā pieeja LLM veidošanai ir izmantot uz transformatoriem balstītus modeļus.

> ***Transformators (Transformer)*** *ir revolucionāra arhitektūra, kas ieviesta rakstā "Attention is All You Need" autorā Vaswani et al. (2017). Kopš tā laika tā ir kļuvusi par pamata arhitektūru daudziem NLP (dabiskās valodas apstrādes, Natural Language Processing) uzdevumiem, ieskaitot teksta ģenerēšanu.*

Tomēr Jānis zina par transformatoru modeļu trūkumu: tie ir autoregresīvi savā būtībā, kas ierobežo teksta ģenerēšanas ātrumu un efektivitāti.

> ***Autoregresīviem modeļiem (Autoregressive models)*** *tokeni ir jāģenerē pa vienam secībā: pēc tam, kad tokens ir ģenerēts, to pievieno rezultātam, un nākamo tokenu ģenerē, pamatojoties uz atjaunināto rezultātu. Tas ierobežo ģenerēšanu uz secīgu.*

Jānis vēlas izpētīt **difūzijas modeļus (Diffusion models)**, kas var ģenerēt tokenus patvaļīgā secībā ātrākai, paralēlai secinājumu veikšanai.

> ***Difūzijas modeļi (Diffusion models)*** *ir neautoregresīvi un var ģenerēt tokenus jebkādā secībā, ne tikai no kreisās uz labo.*

Tomēr Jānis zina par problēmu ar difūzijas modeļiem — tie bieži sabrūk autoregresīvā ģenerēšanā: ģenerē tokenus secīgi no kreisās uz labo. Lai to novērstu, Jānis ierobežo ģenerēšanas secību: vispirms ģenerēt tokenus, kas nav blakus, pēc tam pārējos.

Pirms faktiski izveidot difūzijas LLM, Jānis vēlas prototipēt un izmērīt, cik efektīvāks tas ir.

Viņam ir īpaša GPU, kas patērē vairāk enerģijas, jo augākas tokenu vērtības uz ievades un izvades. Lai minimizētu izmaksas, viņš vēlas aprēķināt **uzkrātās enerģijas izmaksas** (kā aprakstīts zemāk).

Jānis izdomāja šādu problēmu:

Uzdevums

Mums ir n tokenu secība: $P = [p_1, p_2, \ldots, p_n]$, kur katram tokenam p_i ir enerģijas izmaksas, kas vienādas ar tā vērtību. Jāatrod optimālā ģenerēšanas secība, kas minimizē **uzkrātās enerģijas izmaksas**, vienlaikus novēršot **autoregresīvo sabrukumu**.

Uzkrātās enerģijas izmaksas secībai $[a_1, a_2, \ldots, a_n]$ ir:

```
sum = a₁ + (a₁ + a₂) + (a₁ + a₂ + a₃) + ... + (a₁ + a₂ + ... + aₙ)
```

Piemēram, $[1, 3, 9]$ ir uzkrātās izmaksas $= 1 + (1+3) + (1+3+9) = 18$

Ierobežojums, lai novērstu **autoregresīvo sabrukumu**:

- Sadaliet tokens divās grupās pēc to pozīcijas paritātes oriģinālajā secībā: nepāra pozīcijas (1., 3., 5., ...) un pāra pozīcijas (2., 4., 6., ...)
- Visi tokens no vienas paritātes grupas jāizvada pirms jebkuriem tokens no otras grupas

Izvadei jābūt sākotnējās secības P permutācijai.

Ievade

Pirmais arguments ir vesels skaitlis n ($1 \leq n \leq 10^5$) — secības garums.

Otrais arguments ir n ar atstarpi atdalīti veseli skaitļi $p_1, p_2, \ldots, p_n$ ($1 \leq p_i \leq 10^9$) — secības elementi.

Izvade

Izvadiet n ar atstarpi atdalītus veselus skaitļus, kas reprezentē optimālo derīgo permutāciju.

Ja pastāv vairāki optimāli risinājumi, varat izvadīt jebkuru no tiem.

Ierobežojumi

- Testa gadījumu skaits ≤ 100
- n summa visiem testa gadījumiem $\leq 5 \times 10^5$
- Izpildes laika ierobežojums: 5 sekundes
- Atmiņas ierobežojums: 512 MB

Piemēri

Ievads:

```
3
5 1 4
```

Izvads:

```
1 4 5
```

Šī atbilde seko maiņas rakstam: vispirms izvēloties žetonu pāra pozīcijā (2.: vērtība 1), pēc tam žetonus nepāra pozīcijās (1. un 3.: vērtības 5 un 4, sakārtotas kā 4, 5). Uzkrātās izmaksas ir 1 + (1+4) + (1+4+5) = 16, kas ir optimāli.

Vērtēšana

Punkti = nokārtotie testi / kopējie testi

Tālāka izpēte

Ja Jūs interesē, kā darbojas LLM, apskatiet šo video no 3Blue1Brown: http://youtube.com/watch?v=LPZh9BOjkQs

Ja Jūs vēlaties uzzināt vairāk, turpiniet ar viņa neironu tīklu sēriju pēc LLM video! Atskaņošanas sarakstā ir viesa video no Welch Labs par to, kā darbojas transformatori: https://youtube.com/playlist?list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi