# იპოვეთ თანმიმდევრობა

- **დროის ლიმიტი:** 10 წუთი
- **გარემო:** ერთი GPU (≈16 GB VRAM), ინტერნეტის გარეშე
- **ამოხსნის ზომა:** `solution.ipynb` ≤ 1 MB
- **მეხსიერება:** 5 GB

## ამოცანა

თქვენ გეძლევათ ინგლისურენოვანი ზეპირი დიალოგები ორ მონაწილეს (*Speaker A* და *Speaker B*) შორის. თითოეული დიალოგი დაყოფილია მოლაპარაკის სვლებად (speaker turns), სადაც თითოეული სვლა შეიცავს მხოლოდ ერთი მოლაპარაკის მეტყველებას. თითოეული სვლა შენახულია ცალკეულ `.wav` აუდიო ფაილად, ასე რომ, სრული დიალოგი წარმოდგენილია `.wav` ფაილების სიმრავლით, თითო ფაილი თითოეული სვლისთვის.

სამწუხაროდ, სვლები შემთხვევითად არის არეული, რის გამოც საუბარი აზრს კარგავს. ფაილის სახელში `chunk_{k}.wav`, `k` აღნიშნავს არეულ სიმრავლეში k-ურ ფრაგმენტს (chunk) და არა k-ურ სვლას ორიგინალურ დიალოგში.

**!! თქვენი ამოცანაა აღადგინოთ საუბრის ორიგინალური ქრონოლოგიური თანმიმდევრობა.**

# მონაცემთა სიმრავლე

თითოეული დიალოგი შეიცავს `n` აუდიო ფაილს სახელებით `chunk_0.wav`, `chunk_1.wav`, …, `chunk_{n-1}.wav`. ფრაგმენტები წარმოადგენს ცალკეულ სვლებს. ფაილის სახელები შეესაბამება მხოლოდ არეულ თანმიმდევრობას. ისინი არ მიუთითებენ, თუ სად მდებარეობს ფრაგმენტი ორიგინალ საუბარში. თითოეულ დიალოგს აქვს 7–20 ფრაგმენტი, მონო, 44.1 კჰც (შეგიძლიათ შეცვალოთ დისკრეტიზაციის სიხშირე - resample).

**`prefix.json` შეიცავს თითოეულ დიალოგში პირველი ორი რეალური ფრაგმენტის ფაილის სახელის ინდექსებს.** ეს განსაზღვრავს დიალოგის ჭეშმარიტ დასაწყისს და ხსნის გაურკვევლობას საუბრის წინიდან ან უკნიდან გააზრებას შორის.

მაგალითად: `11: [7, 12]` ნიშნავს, რომ მე-11 დიალოგის პირველი და მეორე სვლები არის შესაბამისად `chunk_7.wav` და `chunk_12.wav`.

## რას იღებთ თქვენ

თქვენ იღებთ **ორ საქაღალდეს იდენტურ ფორმატში**:

| საქაღალდე | დიალოგები | `answers.json`? | გამოიყენეთ ის, რათა |
|---|---|---|---|
| `dataset/train/` | 1,288 | ✅ მოცემულია | გაწვრთნათ / დაატუნინგოთ (fine-tune) თქვენი მოდელი |
| `dataset/test_public/` | 100 | ✅ მოცემულია | გაუშვათ თქვენი ფაიფლაინი (pipeline) და ლოკალურად შეაფასოთ საკუთარი შედეგი |

შეფასებისას, თქვენი `dataset/test_public/` საქაღალდე გამჭვირვალედ იცვლება `hidden evaluation set` საქაღალდით (`test_leaderboard_a` საჯარო ლიდერბორდისთვის (leaderboard) და `test_leaderboard_b` საბოლოო ლიდერბორდისთვის) — მათ აქვთ იგივე ზომა და ფორმატი, რაც `dataset/test_public/`-ს, მაგრამ `answers.json`-ის გარეშე.

თქვენი ნოთბუქი (notebook) ხელახლა გაეშვება ამ მონაცემებზე, ხოლო მის მიერ გენერირებული `answers.json` ფაილი გამოყენებული იქნება შეფასებისთვის. სატესტო დიალოგები მოდის იმავე განაწილებიდან, საიდანაც `train`, ასე რომ თქვენი ლოკალური `test_public` ქულა სანდო წინასწარი შეფასებაა.

### ფოლდერების სტრუქტურა

```
dataset/train/
    prefix.json  # {dialogue_id: [first_idx, second_idx]} filename index
    answers.json  # {dialogue_id: P}  ground-truth order (rank convention)
    /
        chunk_0.wav
        ...
        chunk_{n-1}.wav

dataset/test_public/
    prefix.json
    answers.json     # present only in the development copy
    /
        chunk_0.wav
        ...
        chunk_{n-1}.wav
```

---

## აუთფუთის მონაცემები

თითოეული დიალოგისთვის განსაზღვრეთ მისი აუდიო ფრაგმენტების ორიგინალური ქრონოლოგიური თანმიმდევრობა. თქვენი პროგნოზი უნდა იყოს `{0, 1, …, n−1}`-ის პერმუტაცია (permutation) `P`, სადაც `P[i]` არის `chunk_i.wav`-ის პროგნოზირებული ქრონოლოგიური პოზიცია (0 = პირველი).

თქვენი გამოსავალი ფაილი `answers.json` თითოეული დიალოგის ID-ს უნდა უთითებდეს მის პროგნოზირებულ პერმუტაციას:

```
{
  "17": [2, 0, 1],
  "42": [0, 1],
  "108": [3, 1, 0, 4, 2, 5]
}
```

### მაგალითი

დიალოგს აქვს 3 არეული ფრაგმენტი `chunk_0, chunk_1, chunk_2`:

| არეული ფრაგმენტი | ნათქვამი შინაარსი | ჭეშმარიტი პოზიცია (რანგი) |
|---|---|---|
| `chunk_0.wav` | *"No worries — I'll send you the notes afterwards."* | 2 (ბოლო) |
| `chunk_1.wav` | *"Hey, are you coming to the three o'clock meeting?"* | 0 (პირველი) |
| `chunk_2.wav` | *"I can't — I've got a dentist appointment then."* | 1 |

ჭეშმარიტი თანმიმდევრობაა **chunk_1** → **chunk_2** → **chunk_0**, შესაბამისად `P = [2, 0, 1]`, ხოლო `prefix.json` შეიცავს `[1, 2]`-ს.

⚠ **P უნდა იყოს ნამდვილი პერმუტაცია:** სიგრძე n, 0-ზე დაფუძნებული ინდექსაციით (0-indexed), თითოეული მნიშვნელობა ზუსტად ერთხელ. დუპლიკატები, ნაკლული მნიშვნელობები ან დიაპაზონს გარეთ არსებული ჩანაწერები (მაგ. 1-ზე დაფუძნებული ინდექსაციით) ამ დიალოგისთვის ფასდება 0 ქულით, ისევე როგორც ფაილიდან დანაკლისი დიალოგი. არასწორი ფორმატის ან non-JSON ფაილი უარყოფილი იქნება.

## შეფასება

ამ ამოცანის შეფასება ეფუძნება **წყვილური თანმიმდევრობის სიზუსტეს (pairwise ordering accuracy)**. ის ამოწმებს ფრაგმენტების თითოეულ წყვილს და სვამს კითხვას: *რომელი უნდა იყოს პირველი ამ ორს შორის?* წყვილი სწორია, თუ თქვენი პროგნოზი იძლევა იმავე პასუხს, რასაც რეალური ჭეშმარიტება (ground truth). `n` ფრაგმენტის მქონე დიალოგისთვის არსებობს

$$M = n(n-1)/2$$

წყვილი; მოდი `I` იყოს ინვერსიების რაოდენობა — ისეთი წყვილების, რომლებიც დალაგებულია რეალური ჭეშმარიტებისგან განსხვავებულად:

$$\text{score} = 1 - \frac{I}{M} \in [0, 1]$$

i **საბოლოო ქულა არის დიალოგების მიხედვით მიღებული ქულების საშუალო მონაცემთა ნაკრების (split) ყველა დიალოგზე.**

## დაშვებული მოდელები

ამ ამოცანის გადასაჭრელად შეგიძლიათ გამოიყენოთ მხოლოდ შემდეგი წინასწარ დატრენინგებული (pre-trained) მოდელები, როგორც წვრთნის, ასევე შეფასების ეტაპზე. ყველა ეს მოდელი უკვე ჩამოტვირთულია და ხელმისაწვდომია გარემოში. მათი გამოყენების მაგალითები შეგიძლიათ იხილოთ საბაზისო ნოთბუქში `solution.ipynb`. გთხოვთ, გაითვალისწინოთ, რომ არ შეგიძლიათ სხვა მოდელის გამოყენება და თქვენს პროგრამას არ აქვს წვდომა ინტერნეტთან.

- **აუდიო რეპრეზენტაციები (Speech representations): wav2vec 2.0**. **Whisper encoder**-ც ასევე შეიძლება გამოყენებულ იქნას როგორც ფიჩერ ექსტრაქტორი (feature extractor). [wav2vec model card]

- **მეტყველების ავტომატური ამოცნობა (ASR): OpenAI Whisper** (ნებისმიერი ზომის). [Whisper model card]

- **ენობრივი მოდელი (Language model): Qwen2.5-0.5B**, რომელიც შეიძლება გამოყენებულ იქნას როგორც zero-shot რეჟიმში, ასევე დატუნინგებული (fine-tuned) მოწოდებულ `train` მონაცემთა ნაკრებზე. [Qwen model card] გაითვალისწინეთ, რომ 10-წუთიანი ლიმიტი უნდა ფარავდეს ნებისმიერ წვრთნასა თუ დატუნინგებას (fine-tuning), რომელსაც შეფასებისას აკეთებთ, პლუს ინფერენსს (inference) შეფასების სიმრავლეზე.

## როგორ გააგზავნოთ ნამუშევარი

- გახსენით `solution.ipynb` და გააშვით ყველა უკჯრა (cell). დარწმუნდით, რომ ის სამუშაო დირექტორიაში წერს `answers.json` ფაილს, პერმუტაციით `dataset/test_public/`-ში არსებული თითოეული დიალოგისთვის (100 დიალოგი). შეფასებისას ნოთბუქი ხელახლა გაეშვება ფარულ სატესტო სიმრავლეზე და შეფასდება იქვე გენერირებული `answers.json`.
- გააუმჯობესეთ ამოხსნა, თუ გსურთ — ან არ გააუმჯობესოთ; ბეიზლაინი (baseline) მხოლოდ ადასტურებს ფაიფლაინის მუშაობას.
- გახსენით Git ტაბი JupyterLab-ის მარცხენა პანელზე.
- **დაასტეიჯეთ (Stage)** `solution.ipynb` (მის გვერდით არსებული + იკონით).
- შეიყვანეთ კომიტის შეტყობინება (commit message) და დააჭირეთ **Commit**-ს.
- დააჭირეთ ღრუბლისა და ზემოთ მიმართული ისრის იკონს გასაგზავნად (push).
- დაბრუნდით კონკურსის (Contest) ამ გვერდზე და დააჭირეთ **Submit**-ს.

გააგზავნეთ ზუსტად ერთი ფაილი, სახელად `solution.ipynb`.