Laboratorium 6 — TRM

Wojciech Bartoszek

Streszczenie

Niniejszy raport dokumentuje projekt, trening oraz ocenę Tiny Recursive Model (TRM) zastosowanego do zadania Sudoku Extreme oraz porównuje jego zachowanie z pojedynczym modelem LLM ("ministral-3:3b") przy użyciu dwóch strategii promptowania: bezpośredniej ("zero") oraz łańcucha myślenia ("cot"). Przedstawiono krzywe trenowania, metryki ilościowe oraz jakościową analizę błędów obserwowanych w odpowiedziach LLM.

1 Zadanie i cele eksperymentu

Celem zadania było sprawdzenie, czy niewielki model o indukcyjnych priorytetach wnioskowania może nauczyć się rozwiązywać strukturalne zadania rozumowania (tutaj: Sudoku 9x9). Waga została położona na jakość wnioskowania, ekonomię danych i analizę błędów, a nie na skalę modelu.

2 Dane i wstępne przetwarzanie

Użyto dostarczonego zbioru Sudoku Extreme (z próbkowaniem i augmentacją, aby zapewnić wykonalność treningu). Metadane: długość sekwencji 9×9 (seq_len = 81), kodowanie słownika opiera się na liczbach całkowitych, gdzie 0 oznacza puste pole; tokeny wyjściowe modelu były przesunięte o +1 wewnętrznie. Dane zostały przygotowane i zapisane w `lab6/data/sudoku-extreme-enhanced`. Zbiór treningowy został próbkowany do 10k przykładów, by trening był wykonalny na dostępnych zasobach sprzętowych.

3 Architektura modelu i trening

Model. TRM to mały model implementujący mechanizm zatrzymania w stylu ACT. W przeprowadzonym eksperymencie zastosowano następującą konfigurację:

- Rozmiar warstwy ukrytej: **128**
- Liczba głów (heads): **4**
- Liczba warstw lokalnych (L-layers): **2**
- Maksymalna liczba kroków zatrzymania: **4**
- Rozmiar batcha: **32**
- Optymalizator: AdamW (lr = 1×10^{-3}, weight_decay = 0.01)
- Harmonogram: ReduceLROnPlateau (factor=0.5, patience=5)
- Liczba epok treningu: **5**

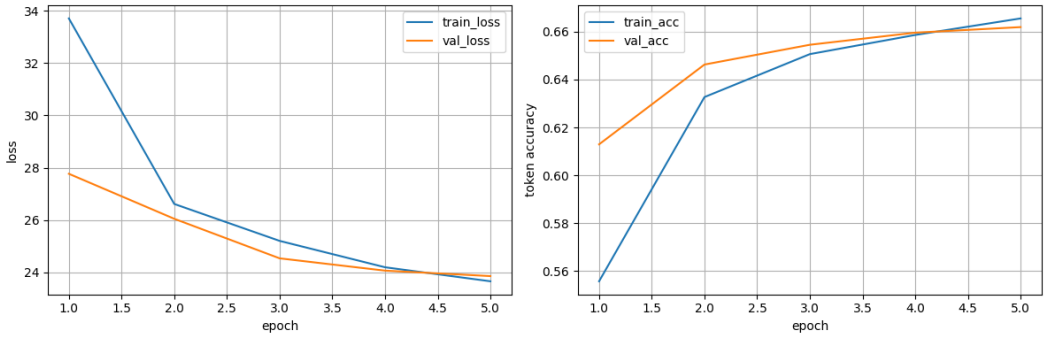
4 Wyniki trenowania

Tabela 1 zawiera metryki rejestrowane w kolejnych epokach treningu.

Tabela 1: Metryki treningowe i walidacyjne na epokę

Epoka	train_loss	train_token_acc	val_loss	val_token_acc
1	33.702	0.556	27.769	0.613
2	26.617	0.633	26.051	0.646
3	25.207	0.651	24.540	0.655
4	24.200	0.659	24.071	0.660
5	23.664	0.666	23.863	0.662

Rysunek 1 przedstawia krzywe strat i dokładności tokenowej dla treningu i walidacji (plik: `lab6/checkpoints/training_history.png`).



Rysunek 1: Historia treningu: strata (po lewej) i dokładność tokenowa (po prawej).

Końcowa dokładność tokenowa na zbiorze walidacyjnym dla TRM wynosi **0.6619** (patrz Tabela **??**), a strata walidacyjna to **23.8626**.

5 Ocena modelu LLM (ministral-3:3b)

Dla porównania z TRM oceniono pojedynczy, silny model LLM na niewielkim zbiorze przykładów (5 zadań testowych) używając dwóch strategii:

- **zero**: bezpośredni prompt żądający jedynie finalnej, numerycznej reprezentacji siatki.
- **cot**: prośba o krótkie rozumowanie krok-po-kroku (chain-of-thought), przy czym model ma zwrócić tylko finalną siatkę.

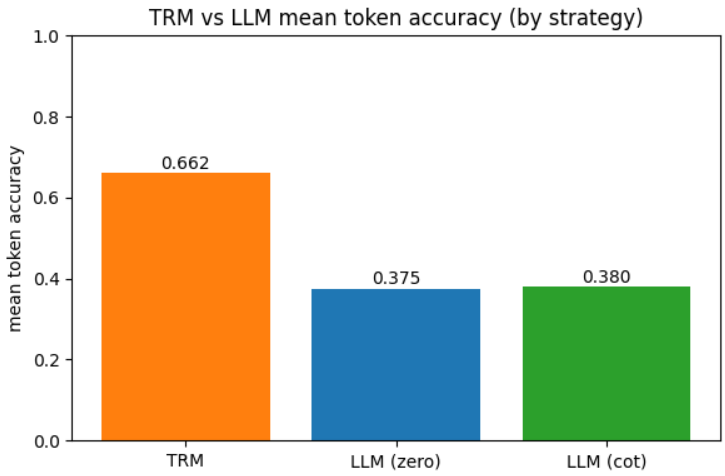
Interakcja odbyła się za pośrednictwem lokalnego klienta `ollama`, a odpowiedzi były parsowane heurystycznie (wyszukiwanie bloków kodu lub sekwencji cyfr).

Tabela 2 przedstawia dokładności tokenowe dla pojedynczych przykładów oraz średnie dla każdej strategii.

Wykres porównujący TRM oraz dwie strategie promptowania LLM zapisano w `lab6/checkpoints/trm_vs_llm_` i pokazano na Rysunku 2.

Tabela 2: Tokenowa dokładność LLM dla poszczególnych przykładów (5 przykładów testowych)

Przykład	zero (token_acc)	cot (token_acc)
0	0.567 901	0.518 519
1	0.407 407	0.506 173
2	0.370 370	0.259 259
3	0.283 951	0.308 642
4	0.246 914	0.308 642
średnia	**0.375309**	**0.380247**



Rysunek 2: Średnia dokładność tokenowa: TRM vs LLM (zero) vs LLM (cot).

6 Analiza jakościowa

TRM nauczył się spójnego wzorca rozwiązywania na poziomie tokenów (końcowa dokładność 0.662), natomiast LLM wykazał następujące zachowania:

- Problemy z formatem wyjścia: wiele odpowiedzi LLM zawierało powtarzające się cyfry, zduplikowane wiersze oraz wartości łamiące reguły Sudoku (np. powtórzenia w wierszu), co czyniło rozwiązania niepoprawnymi.
- Kruchość parsowania: mimo zastosowania heurystyk parsujących (wyodrębnianie sekwencji cyfr i sprawdzanie zakresu/kształtu), część odpowiedzi LLM wymagała czyszczenia lub była odrzucona (zarejestrowane jako błędy parsowania).
- Wpływ promptowania: zastosowanie "chain-of-thought"("cot") nie poprawiło jednoznacznie dokładności tokenowej względem prostego promptu ("zero") na tej małej próbce (średnie: 0.380 vs 0.375).

Obserwacje te zgadzają się z oczekiwaniem, że model ogólnego przeznaczenia bez jawnego wymuszania reguł Sudoku może generować odpowiedzi syntaktycznie prawdopodobne, ale niezgodne z ograniczeniami problemu.

7 Dyskusja i ograniczenia

- TRM wykazuje solidną jakość na poziomie tokenów w porównaniu z LLM w trybie few-shot, co podkreśla zalety modelu specjalizowanego do zachowań strukturalnych i spójności lokalnej.
- Ocena oparta była na dokładności tokenowej (porównanie przewidzianych tokenów komórek z prawdziwymi). Bardziej rygorystyczna metryka — poprawne kompletne rozwiązanie Sudoku — byłaby bardziej informatywna dla zastosowań praktycznych.
- Ocena LLM była ograniczona do niewielkiej próbki 5 przykładów i jednego modelu/ustawienia; wyniki nie są wystarczające do szerokich uogólnień.
- Część odpowiedzi LLM była sformatowana niepoprawnie; lepszy pipeline ewaluacyjny mógłby zawierać solver weryfikujący spójność i mierzący poprawność całej siatki oraz spełnianie ograniczeń.

8 Wnioski

Specjalizowany TRM trenowany end-to-end na zadaniu Sudoku osiąga istotnie wyższą dokładność tokenową niż pojedynczy duży LLM oceniany na małej próbce przy dwóch strategiach promptowania. Małe modele z odpowiednimi priorytetami inductywnymi są skuteczne w zadaniach strukturalnych, gdzie ich założenia i funkcja celu odpowiadają ograniczeniom domeny.