Porównanie Modeli LSTM i Transformer w Zadaniu Modelowania Języka Polskiego

Łukasz Chęciak

26 października 2025

Streszczenie

Niniejszy raport przedstawia proces implementacji, treningu oraz porównania dwóch architektur sieci neuronowych – LSTM (Long Short-Term Memory) i Transformer – w zadaniu modelowania języka. Celem projektu było zbadanie różnic w jakości generowanego tekstu oraz wydajności obu modeli. (trening na CPU).

1 Opis Zbioru Danych

Do treningu modeli wykorzystano zbiór danych **Speakleash**, a konkretnie jego część `wolne_lektury_corpus`, która zawiera teksty polskich lektur szkolnych. Aby umożliwić realizację projektu na sprzęcie bez dedykowanej karty graficznej, zbiór został ograniczony do pierwszych **1000 dokumentów**.

1.1 Proces Przygotowania Danych

1. **Tokenizacja:** Cały korpus został poddany tokenizacji przy użyciu podstawowego tokenizatora (‘basic_english‘ z biblioteki `torchtext`), który dzieli tekst na słowa w oparciu o spacje i znaki interpunkcyjne.
2. **Budowa Słownika:** Na podstawie tokenów zbudowano słownik, który zmapował unikalne słowa na indeksy liczbowe. Rozmiar słownika wyniósł **6255038** tokenów.
3. **Podział na Zbiory:** Dane podzielono w proporcjach: 90% na zbiór treningowy, 5% na walidacyjny i 5% na testowy.

2 Architektura Modeli

Oba modele zostały zaimplementowane w bibliotece PyTorch. Poniżej przedstawiono ich kluczowe parametry.

2.1 Model LSTM

Sieć rekurencyjna o prostej architekturze, przetwarzająca dane sekwencyjnie.

- **Warstwa wejściowa:** `nn.Embedding` o rozmiarze wektora 200.
- **Warstwy ukryte:** Dwie warstwy `nn.LSTM` z 200 neuronami w stanie ukrytym.
- **Warstwa wyjściowa:** `nn.Linear` mapująca stan ukryty na rozmiar słownika.
- **Dropout:** 0.2.
- **Optymalizator:** `SGD` z krokiem uczenia 5.0 i schedulerem `StepLR` (gamma=0.95).

2.2 Model Transformer

Architektura oparta na mechanizmie uwagi, zdolna do równoległego przetwarzania sekwencji.

- **Warstwa wejściowa:** `nn.Embedding` (rozmiar 200) oraz kodowanie pozycyjne.
- **Warstwy enkodera:** Dwie warstwy `nn.TransformerEncoderLayer` z 2 głowicami uwagi (`nhead`) i 200 neuronami w warstwie FFN.
- **Warstwa wyjściowa:** `nn.Linear` mapująca wyjście enkodera na rozmiar słownika.
- **Dropout:** 0.2.
- **Optymalizator:** Identyczny jak w modelu LSTM.

3 Wyniki Ewaluacji

Trening obu modeli ograniczono do około **7 godzin** na CPU. Metrykami oceny były perpleksja (im niższa, tym lepiej) oraz czas treningu.

3.1 Metryki Wydajności

Metryka	Model LSTM	Model Transformer
Strata walidacyjna	8.0	7.53
Perpleksja (PPL)	2789.79	1859
Średni czas / paczkę	∼3.3 s	∼2.8 s

Tabela 1: Porównanie metryk końcowych po ∼7 godzinach treningu.

3.2 Porównanie Wygenerowanych Tekstów

Po treningu oba modele zostały użyte do wygenerowania kontynuacji dla 10 identycznych promptów. Wyniki zostały zapisane w folderze `results`. Poniżej przykładowe porównanie dla promptu: *"Dawno, dawno temu za siedmioma górami"*.

3.2.1 Wynik LSTM

"dawno , dawno temu za siedmioma górami , żeby z nim jego w istocie nie zechce . ja jestem tak nie ? gdzie wszakże umarł z dwóch literatury . tak jak mówiłem ? i tak się trzeba pan . kiedy położył się do niej z ludzi i ziemi , żeby jęli się antyli w imieniu co potrzeba"

3.2.2 Wynik Transformer

"dawno , dawno temu za siedmioma górami do nogi ? — powtórzył cofnął się nie chce była świt , który na tę historię colombo przez okienko , który inny , tak że nie było wreszcie okna , ale nie zapomnę się ino do nauki uśmiéchnąć . nikt nie byłoby z góry , nagle szedł przy sobie coś"

3.3 Interpretacja

Interpretacja: Model LSTM wykazuje tendencję do generowania tekstu spójnego gramatycznie, ale często wpada w pętle lub traci główny wątek. Z kolei Transformer, dzięki mechanizmowi uwagi, lepiej utrzymuje kontekst, co prowadzi do bardziej zróżnicowanych i logicznych kontynuacji, nawet jeśli lokalnie tekst bywa mniej płynny.

4 Wyzwania Implementacyjne i Wnioski

- **Wydajność CPU:** Największym wyzwaniem był bardzo długi czas treningu. Ograniczenie zasobów obliczeniowych uniemożliwiło pełne wytrenowanie modeli, co wpłynęło na ich ostateczną jakość.
- **Zarządzanie procesem treningu:** Aby uniknąć utraty postępów, zaimplementowano mechanizm **checkpointingu**, który cyklicznie zapisywał stan modelu i optymalizatora. Pozwoliło to na bezpieczne przerywanie i wznawianie długotrwałego procesu uczenia.

4.1 Wnioski Końcowe

Projekt z powodzeniem zademonstrował proces budowy i porównania dwóch kluczowych architektur w dziedzinie NLP. Mimo ograniczeń sprzętowych, udało się zaobserwować fundamentalne różnice w ich działaniu. Model Transformer pokazał swój potencjał w generowaniu bardziej kontekstowych i interesujących treści. Model LSTM okazał się prostszy i szybszy w przeliczeniu pojedynczej paczki, ale jego "pamięć"była wyraźnie krótsza. Eksperyment potwierdza, że do zaawansowanych zadań językowych architektury oparte na mechanizmie uwagi są obecnie standardem branżowym.