Lingwistyka Obliczeniowa - Laboratorium 2

Analiza Efektywności Tokenizacji

Andrzej Szarata

Listopad 2025

Link do repozytorium GitHub

1. Opis zbioru danych

Eksperyment przeprowadzono na danych pozyskanych z serwisu *Wolne Lektury*. Zbiór treningowy obejmował teksty autorstwa **Henryka Sienkiewicza**, natomiast zbiór testowy stanowiły utwory **Zofii Nałkowskiej**, niewykorzystywane podczas uczenia modeli.

2. Sprzęt i środowisko

Trening i ewaluację przeprowadzono na komputerze wyposażonym w:

- Apple M4 Pro,
- 20-rdzeniowe GPU (akcelerator MPS),

Środowisko programistyczne:

- Python 3.13,
- PyTorch (akceleracja MPS),
- tokenizers.

3. Tokenizatory

W eksperymencie przygotowano trzy niezależne tokenizatory, różniące się strategią segmentacji i zakresem słownika. Wszystkie wykorzystują słownik o rozmiarze **50 000 tokenów**.

3.1. Tokenizator pretrained (Polish SPLADE)

Pierwszy tokenizator stanowi gotowy, publicznie dostępny model:

- **sdadas/polish-splade** (HuggingFace),
- zbudowany w oparciu o algorytm BPE,
- zawierający reguły dostosowane do języka polskiego.

Tokenizator został pobrany i zapisany na urządzeniu w formie pliku `.json` kompatybilnego z biblioteką `tokenizers`.

3.2. Tokenizator Whitespace

Drugi tokenizator implementuje klasyczną segmentację opartą na białych znakach. Cechy:

- rozdzielanie po spacji oraz znakach interpunkcyjnych,
- słownik tworzony na podstawie częstości słów,
- rzadkie słowa mapowane na token `<unk>`,
- brak podziału słów: każde słowo → dokładnie jeden token.

Implementacja opiera się o `WordLevel` z biblioteki `tokenizers`. Użyto specjalnych tokenów: `<pad>`, `<s>`, `</s>`, `<unk>`, `<mask>`.

3.3. Tokenizator BPE (trening własny)

Trzeci tokenizator wytrenowano z użyciem algorytmu **Byte-Pair Encoding** na pełnym zbiorze treningowym. Cechy:

- implementacja: `tokenizers` (model BPE),
- pre-tokenizer: byte-level,
- dekoder: byte-level,
- słownik 50 000 elementów,
- minimalna częstość wykrywania par: 2.

Tokenizator ten umożliwia rozbicie rzadkich słów na subtokeny, co znacząco redukuje OOV.

4. Architektura modelu językowego

Dla wszystkich trzech tokenizatorów trenowano identyczny model, implementujący uproszczoną architekturę **GPT-2 Decoder-only**, napisaną samodzielnie w PyTorch.

4.1. Parametry architektury

- maksymalna liczba pozycji: 1024,
- liczba warstw: 8,
- liczba głów atencji: 6,
- rozmiar embeddingów: 768,
- rozmiar warstw ukrytych: 1024,
- dropout: 0.1.

Model uczono autoregresyjnie, przewidując kolejny token sekwencji.

5. Parametry treningu

- batch size: 64,
- liczba epok: 6,
- długość sekwencji: 128,
- optymalizator: AdamW,
- learning rate: 5×10^{-5}.

Czas treningu dla poszczególnych tokenizatorów:

Tokenizator	**Czas treningu**
Pretrained (SPLADE)	53 min 28 s
BPE (trening własny)	37 min 50 s
Whitespace	25 min 09 s

6. Wyniki eksperymentów

W tej sekcji przedstawiono:

- word-level perplexity,
- character-level perplexity,
- OOV rate,
- średnią liczbę tokenów na słowo,
- liczbę słów obecnych w słowniku tokenizatora,
- przepustowość tokenizatora (tokens/s).

6.1. Perplexity

Tokenizator	**PPL (word)**	**PPL (char)**	**PPL (token)**
Pretrained (SPLADE)	1.5396	1.5391	1.5415
BPE	1.8916	1.8922	1.8913
Whitespace	2.2989	2.3010	2.3044

6.2. OOV rate

Dla tokenizatora **Whitespace** wyliczono statystykę **OOV** (Out of Vocabulary). Wynik wyniósł **20.56%**.

6.3. Przepustowość tokenizatora (tokens/s)

Tokenizator	Throughput
BPE	1 335 896.57
Pretrained (SPLADE)	657 917.84
Whitespace	575 230.53

6.4. Średnia liczba tokenów na słowo

Tokenizator	Tokens/word
Whitespace	1.0000
BPE	1.5102
Pretrained (SPLADE)	1.5679

6.5. Liczba słów w słowniku

Tokenizator	W słowniku	Procent
Whitespace	85756	79.44%
Pretrained (SPLADE)	38827	35.97%
BPE	32061	29.70%

7. Wnioski

Analiza wyników eksperymentu wskazuje, że wybór tokenizatora wpływa zarówno na jakość reprezentacji tekstu, jak i na efektywność trenowanego modelu językowego.

- **Tokenizator pretrained (SPLADE)** osiągnął najniższe wartości perplexity (prawdopodobnie błędne; 1.5396 dla word-level, 1.5391 dla character-level), co wskazuje na jego zdolność do dokładniejszego modelowania

sekwencji i przewidywania kolejnych tokenów. Wynika to z zaawansowanej segmentacji słów, która pozwala na uchwycenie specyfiki języka polskiego oraz redukcję rzadkości słów w słowniku. Pomimo tego tokenizator ten generuje średnio 1.5679 tokena na słowo, co zwiększa długość sekwencji wejściowej i obciążenie pamięciowe modelu (w tym wyraźnie najdłuższy trening), a jego przepustowość jest umiarkowana (657 917.84 tokens/s), co jest kompromisem między złożonością tokenizacji a szybkością przetwarzania.

- **Tokenizator BPE** cechuje się wysoką przepustowością (1 335 896.57 tokens/s), co wynika z prostoty modelu byte-level oraz niskiego narzutu dekodowania. Perplexity (1.8916 word-level, 1.8922 character-level) jest wyższe niż SPLADE, ale wciąż znacznie niższe niż dla tokenizatora Whitespace. Średnia liczba tokenów na słowo wynosi 1.5102, co wskazuje na umiarkowaną segmentację: rzadkie słowa są dzielone na subtokeny, co zmniejsza ryzyko wystąpienia OOV i umożliwia lepsze dopasowanie modelu do testowego języka.

- **Tokenizator Whitespace** zapewnia najprostszą reprezentację: jedno słowo odpowiada jednemu tokenowi, co skutkuje średnio 1.0 tokenem na słowo i umożliwia najszybszy trening (25 min 09 s). Jednak analiza OOV wskazuje, że aż 20.56% słów w zbiorze testowym nie znajduje się w słowniku, co jest istotnym ograniczeniem w języku polskim o bogatej fleksji. Wysokie wartości perplexity (2.2989 word-level, 2.3010 character-level) pokazują, że brak podziału słów uniemożliwia modelowi skuteczne przewidywanie kolejnych tokenów dla rzadkich lub odmieniających się form słów. Pomimo szybkiego treningu, ograniczona zdolność Whitespace do generalizacji skutkuje gorszą jakością generowanego języka.

- **Porównanie subword vs. word-level**: tokenizatory dzielące słowa (BPE, SPLADE) poprawiają jakość modelu poprzez redukcję problemu OOV oraz lepsze uchwycenie struktury języka polskiego. Dzieje się to kosztem większej liczby tokenów na słowo, co zwiększa złożoność obliczeniową oraz wydłuża czas inferencji.

- **Trade-off między przepustowością a jakością**: BPE zapewnia najwyższą przepustowość, co jest korzystne przy dużych zbiorach danych lub ograniczeniach czasowych, jednak SPLADE oferuje lepsze dopasowanie do języka polskiego i niższe perplexity kosztem niższej wydajności. Whitespace jest szybki i prosty, ale nieefektywny w kontekście jakości generacji.

- **Wnioski praktyczne** Wybór konkretnego rozwiązania zależy od specyfiki zadania: kluczowe jest dostosowanie kompromisu między szybkością przetwarzania a precyzją predykcji. Tokenizator pretrained SPLADE sprawdza się najlepiej, gdy celem jest minimalizacja perplexity i większa generalizacja, natomiast BPE może być preferowany w scenariuszach wymagających dużej przepustowości.