

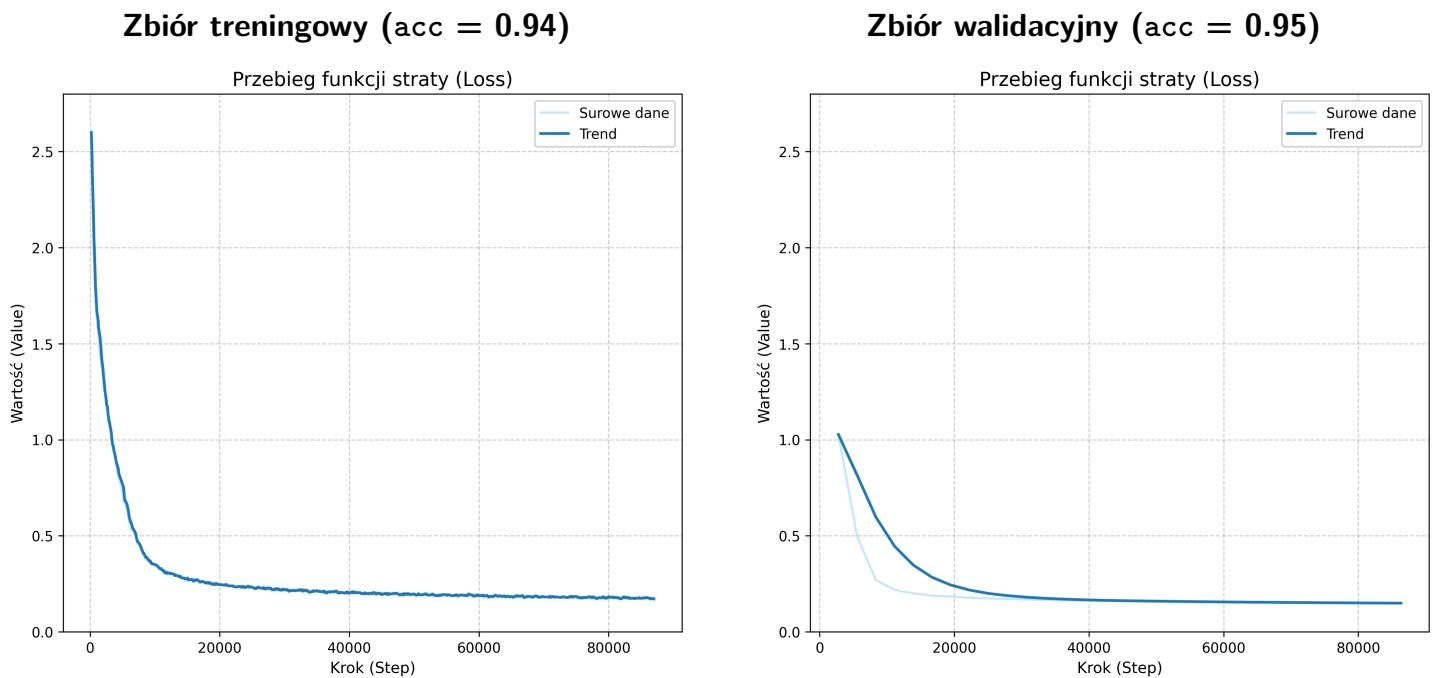
Architektura stworzonego modelu składa się z trzech głównych elementów:

- **warstwa osadzeń (Embedding)**: przekształca tokeny z przestrzeni słownika w wektory reprezentacji,
- **moduł LSTM**: rdzeń rekurencyjny modelu, odpowiedzialny za przetwarzanie informacji sekwencyjnej,
- **warstwa wyjściowa (Linear)**: mapuje wyjście z modułu LSTM z powrotem na przestrzeń słownika.

W celu doboru optymalnych hiperparametrów przeprowadzono serię eksperymentów z następującymi wartościami:

- `embedding_dim` ∈ {64, 128}: mniejsza wartość 64 dawała zdecydowanie gorsze wyniki, wartość **128 (optymalny)** zapewniała lepszą pojemność reprezentacyjną,
- `hidden_dim` ∈ {256, 512}: najlepsze wyniki walidacyjne uzyskano dla **512 (optymalny)**,
- `num_layers` ∈ {2, 4}: optymalna konfiguracja to **4 warstwy (optymalny)**, które osiągały zdecydowanie lepsze wyniki niż 2, przy wyraźnie krótszym czasie treningu,
- `seq_len` ∈ {50, 100, 200}: optymalną długością sekwencji okazała się **100 (optymalny)**, pozwalająca uchwycić zależności kontekstowe między wersami, większa wartość powodowała znaczne wydłużenie treningu bez znaczącego polepszenia,
- `dropout = 0.1 (optymalny)`: wartość zapewniająca skuteczną regularyzację bez pogorszenia jakości predykcji.

Stworzono i wykorzystano znakowy tokenizator, który na podstawie tekstu wyodrębnił unikalne znaki i zbudował z nich słownik obejmujący łącznie 55 tokenów. Trening przeprowadzono z użyciem optymalizatora Adam ($LR = 10^{-3}$) i funkcji straty CrossEntropyLoss w środowisku LightningModule. Proces uczenia nadzorował mechanizm *Early Stopping*, przerywający trening po trzech epokach bez poprawy wyniku walidacyjnego.



Rys. 1: Wykresy przedstawiające spadek wartości funkcji celu w procesie trenowania.

Tabela 1: Wyniki ewaluacji zbioru testowego dla modelu z modułem LSTM

Miara	Wartość	Wartość modelu 128-256
test_acc	0.95	0.91
test_acc_top_5	0.99	0.98
test_ppl	1.16	1.39

Tabela 2: Przykładowe wygenerowane treści dla modelu z modułem LSTM

Prompt	Wynik wybranego modelu	Wynik modelu 128-256
Litwo	litwo przyszło ożenienie, telimena w ich domu miałaby schronienie na przyszłość ; krewna zosi i hrabiego s	litwoi, do zamku progiem do nas była wolno, na zwierza wypuści z podziemu, a zatem zająć robią malowały zr
Główny bohater	główny bohaterów mierzył... że pan jacek, brat mój, tadeusza ocieci, dziwny człowiek, zamiarów jego trudno docieci, ni	główny bohater «pani talerz dobrzy!...» «oj, przerwała — skoczył widny i zdrowie!» wstała młodziana, niepra
po grzyby	po grzyby bracie, trodze skoczył pod parkan ogrodu; tam, pierzchającą rotę zatrzymuje w biegu. szukuje, lecz s	po grzyby głowy położył, a dąsał na krzyż położył na pokrywę wkracza: gdy ponuro, herwawskie lęcze nie можем

Analiza porównawcza wariantów modelu LSTM ujawnia, iż zwiększenie złożoności architektury skutkuje jakościową poprawą generowanych sekwencji, niewidoczną wprost w wartościach klasycznych miar dokładności. Pomimo relatywnie zbliżonych wyników test_acc oraz test_acc_top_5 (tabela 1), wybrany model wyraźnie lepiej radzi sobie z poprawnym tworzeniem słów i budowaniem zdań.

Model o mniejszej pojemności (128–256) częściej generuje błędne słowa oraz ciągi znaków, które nie występują w języku ani w danych treningowych, co wskazuje na trudności w poprawnym odwzorowaniu struktury tekstu. Z kolei architektura o większej liczbie parametrów operuje na pełnych, poprawnie domknietych frazach, które wiernie imitują styl oraz zbiór utworzony z tekstu źródłowego, stwarzając wrażenie poprawności i płynności generowanego tekstu (tab. 2).

Charakterystyczną cechą modelu optymalnego jest zdolność do reprodukcji rozbudowanych fragmentów tekstu, często w postaci całych fraz zaczerpniętych z danych treningowych, choć niepowiązanych z zadanym promptem. Świadczy to o efektywnym przyswojeniu długozasięgowych zależności znakowych, lecz słabszej kontroli ich dopasowania do kontekstu zapytania.

Potwierdzeniem powyższych obserwacji jest wartość miary *perplexity* (tabela 1), która dla wybranego modelu osiąga poziom 1.16, wskazując na minimalną niepewność predykcyjną na etapie wyboru kolejnego znaku. Wyższa PPL wariantu 128–256 (1.39) koreluje bezpośrednio z degradacją jakości tekstu, manifestującą się rozpadem słów i utratą rytmu wypowiedzi. Wyniki te dowodzą, iż w modelowaniu znakowym nawet marginalna redukcja niepewności predykcyjnej ma fundamentalne znaczenie dla końcowej percepcji realizmu generowanej treści.

Różnica między miarami accuracy zbiorów testowego, walidacyjnego (rys. 1) i testowego (tab. 1) wynika z małej różnicy generowanych treści i szablonowego wykorzystania pełnych zwrotów z tekstu źródłowego przez model, niezależnie czy podany prompt był treścią słownikową, czy też nie.

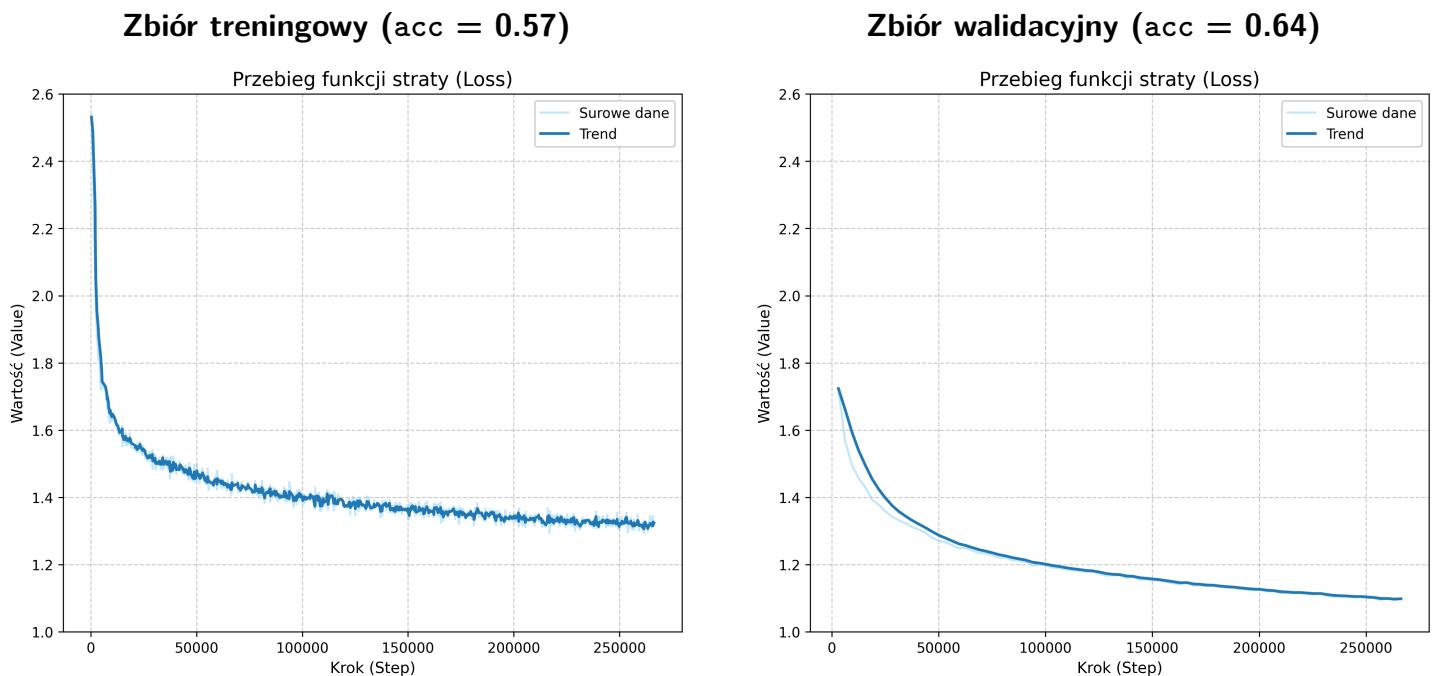
Architektura stworzonego modelu składa się z czterech głównych elementów:

- **warstwa osadzeń (Embedding)**: przekształca tokeny z przestrzeni słownika w wektory z przestrzeni osadzeń,
- **warstwa osadzeń pozycyjnych (PositionalEmbedding)**: dodaje informację o pozycji tokenu do jego reprezentacji wektorowej,
- **moduł TransformerDecoder**: implementuje architekturę dekodera z mechanizmem wielogłówkowej uwagi (Multi-Head Attention),
- **warstwa wyjściowa (Linear)**: mapuje wyjście z dekodera z powrotem na przestrzeń słownika.

W celu doboru optymalnych hiperparametrów przeprowadzono serię eksperymentów z następującymi konfiguracjami:

- $\text{embedding_dim} \in \{64, 128\}$: za najbardziej korzystną uznano wartość **128 (optymalny)**, która zapewnia większą pojemność reprezentacyjną,
- $\text{nhead} \in \{1, 4, 8\}$: najlepszy kompromis między jakością a czasem obliczeń osiągnięto dla **4 (optymalny)**, gdzie model potrafił efektywnie wykorzystywać wielogłowkową uwagę bez istotnego wydłużenia treningu,
- $\text{num_layers} \in \{1, 2, 3\}$: jako konfigurację **optymalną** przyjęto **2 warstwy**, ponieważ 3 warstwy przynosiły poprawę jakości natomiast istotnie zwiększył się czas treningu, a 1 warstwa okazywała się zbyt płytka,
- $\text{seq_len} = \mathbf{100 \ (optymalny)}$: długość sekwencji przejęta z wcześniejszych eksperymentów z LSTM.

Wykorzystany został znakowy tokenizer stworzony w ramach poprzedniego eksperymentu. Trening przeprowadzono z użyciem optymalizatora Adam ($LR = 10^{-3}$) i funkcji straty CrossEntropyLoss w środowisku LightningModule. Nad procesem uczenia czuwał mechanizm *Early Stopping*, przerywający trening po trzech epokach bez poprawy straty walidacyjnej.



Rys. 2: Wykresy przedstawiające spadek wartości funkcji celu w procesie trenowania modelu opartego na TransformerDecoder.

Tabela 3: Wyniki ewaluacji zbioru testowego dla modelu z modułem TransformerDecoder dla pojedynczych znaków

Miara	Wartość	Wartość modelu 1-64
test_acc	0.64	0.48
test_acc_top_5	0.91	0.83
text_ppl	2.99	5.07

Tabela 4: Przykładowe wygenerowane treści dla modelu z modułem TransformerDecoder

Prompt	Wynik	Wynik 1-64
Litwo	litwo, szyk krwią — a jeszcze raz mążny. lecz szerśnicy przebudzić chcesz i te polaków nań prawy; czy jak	litwo podróżny wiem wyjechoby liste usielie operzyjaciele wszerok szej wo w szlachtę dwał rabo już zgromn
Główny bohater	główny bohater usiadł cały, i już wzróciw nie mieszkaniarie na uroczysza: i podniósł dłońca przyka, nie psów hojnie	główny bohater równu: «pan, ściskając snęło imiało, jakrzy wielu mieszaństwa ublilizaniom się, w zisk osy, rozkrzy
po grzyby	po grzyby łowów szlacheckiemych słołach, najł pierwszy zofija króliki ma służyć? zywaliłem to sypać na nim st	po grzybytryumfunkudzą, a na, bogaty, skarwiął tej spodarza pierm: dotąd myślił łowi różnych, tylko bie podsz

W przeciwnieństwie do modeli typu LSTM, architektura TransformerDecoder operująca na poziomie znaków wykazuje podejście czysto generatywne zamiast odtwórczego. Model nie kopiuje sztywnych fragmentów tekstu źródłowego, lecz stara się imitować styl oryginału poprzez predykcję statystycznie prawdopodobnych sekwencji. Efektem tego mechanizmu jest powstawanie słów nieistniejących lub posiadających zmienione końcówki (np. „szerśnicy”, „mieszkaniarie”), które mimo braku poprawności słownikowej, zachowują charakterystyczny dla „Pana Tadeusza” rytm i brzmienie. Większa złożoność modelu bezpośrednio przekłada się na lepszą jakość generacji; podczas gdy model słabszy (1–64) tworzy niemal losowe zbitki liter, model optymalny generuje ciągi, które znacznie częściej układają się w czytelne, choć nie zawsze poprawne gramatycznie wyrazy.

Analiza miar statystycznych zestawionych w Tabeli 3 potwierdza, że wyższa złożoność architektury istotnie redukuje niepewność modelu. Wartość miary test_acc_top_5 na poziomie 0.91 wskazuje, że choć model nie zawsze trafia w konkretny znak (co obniża bazowe test_acc), to właściwa litera niemal zawsze znajduje się w wąskim kręgu pięciu najbardziej prawdopodobnych kandydatów. Świadczy to o dużej różnorodności generowanej treści i zdolności modelu do „rozważania” wielu poprawnych stylistycznie ścieżek kontynuacji tekstu. Potwierdza to również znaczący spadek miary perplexity (PPL z 5.07 do 2.99), co w praktyce oznacza, że model o większej liczbie parametrów podejmuje decyzje znacznie mniej przypadkowo, co widać w spójniejszej strukturze wygenerowanych odpowiedzi w tabeli 4.

Miary accuracy (tab. 3, rys. 2) wskazują na brak przeuczenia modelu, pozwalając na lepsze uogólnienie schematów tekstu treningowego do generowania treści po literach, zachowując pewien stopień ich czytelności.

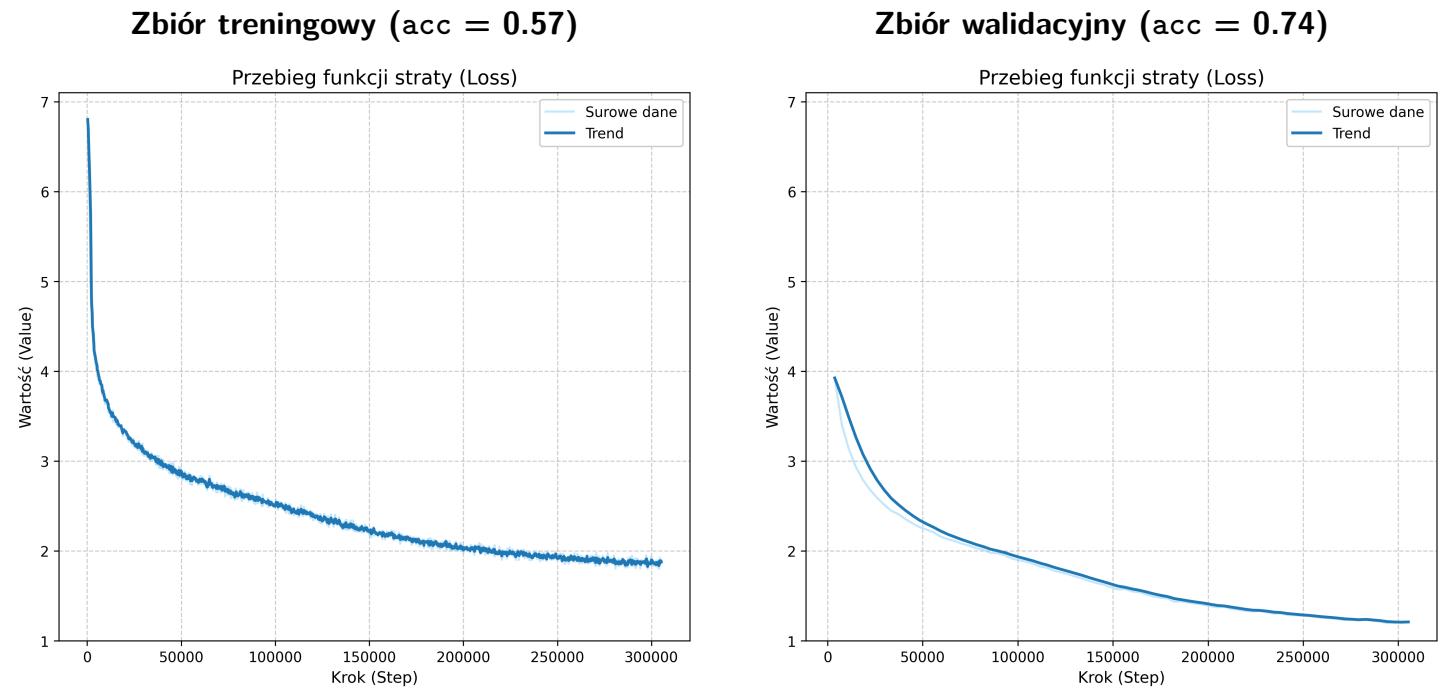
Zastosowano identyczną architekturę jak w poprzednim eksperymencie, jedyną zmianą było użycie innego tokenizera. W celu doboru optymalnych hiperparametrów przeprowadzono serię eksperymentów z następującymi konfiguracjami:

- $\text{embedding_dim} \in \{128, 256\}$: za najbardziej korzystną uznano wartość **128 (optymalny)**, która zapewnia wystarczającą pojemność reprezentacyjną i znaczco przyspieszała trening,
- $\text{nhead} = 4$: na podstawie wyników z poprzedniego eksperimentu wybrano wartość optymalną
- $\text{num_layers} \in \{2, 3\}$: jako konfigurację **optymalną** przyjęto **2 warstwy**, ponieważ 3 warstwy nadmiernie zwiększały złożoność modelu, co prowadziło do pogorszenia jakości.
- $\text{seq_len} = \mathbf{100 \ (optymalny)}$: długość sekwencji przejęta z wcześniejszych eksperymentów.

W tym wariantie rozszerzono również zbiór danych o dwie dodatkowe powieści Henryka Sienkiewicza: *Quo Vadis* oraz *Potop Tom I*.

W tym eksperymencie zastosowano tokenizer oparty na architekturze **GPT-2**, dostosowany do specyfiki użytych tekstów. Na początku tekst jest rozbijany na pojedyncze znaki (bajty), a następnie na tej podstawie budowane są nowe tokeny zgodnie z częstością współwystępowania sekwencji w korpusie. Tokenizer wykorzystuje algorytm *Byte-Level BPE*, który eliminuje problem nierozpoznanych słów, rozbijając je na krótsze, zawsze obsługiwane jednostki (bajty). Przetestowano dwa tokenizery - 5000 i 10000 słów w słowniku. Ostatecznie lepsze wyniki uzyskał tokenizer z większym słownikiem.

Podobnie jak wcześniej, trening przeprowadzono z wykorzystaniem optymalizatora Adam ($LR = 10^{-3}$) oraz funkcji straty CrossEntropyLoss, z użyciem modułu LightningModule. Nad przebiegiem procesu uczenia czuwał mechanizm *Early Stopping*, przerywający trening po trzech epokach bez spadku straty walidacyjnej.



Rys. 3: Wykresy przedstawiające spadek wartości funkcji celu w procesie trenowania modelu opartego na TransformerDecoder z własnym tokenizatorem.

Tabela 5: Wyniki ewaluacji zbioru testowego dla modelu z modułem TransformerDecoder dla własnego tokenizera

Miara	Wartość	Wartość vocab = 5k
test_acc	0.74	0.52
test_acc_top_5	0.88	0.73
test_ppl	3.36	9.65

Tabela 6: Przykładowe wygenerowane treści dla modelu TransformerDecoder z własnym tokenizatorem

Prompt	Wynik	Wyniki vocab = 5k
Litwo	Litwor ciała, widać było ciemne oczy roświeconej boskich i tłumom nie załóż już zgadnie, wszystkie te psem, napełniły za mury zamku, dziątki, trawagi, tobierzeszcze chybały i sentą, gdy w towrzystwie zaniesie w jego bracięgry letyski poznali...» „panie soplico się to z wolna! — przerwał protazy — niech no waść, i zboża	Litwo rzeczy dziewcząć: niemi nic z rzymu. budowle jest wyzwoleniec i tygellinus, sprawdził naprzód, łotrów, który dla którego ręka zatrzyma blade z drugiej strony tybrycjiuszem i dali mu gotowaniem i jego nauzydobicji, by się nie służyć, ale mążny dla zysku, ale prawdzi rzeczpospolita winicjusza. — zatem na ukrainy! — rzekł pan stanisław skrzetuski — tym lepiej... ale on tedy co czynić,
uczelnia	uczelnia skał się od panie, a moja wina, wziąwszy przez którą ci, jako jest bóg błogosławiona, jako ci twoja rzecz... petroniusz, na jego zdrowie moje rysipisz w tym, dlaczego zbrodnia, na którą jaką charakana... nie jest ja wejchne, patrzyłem jak unosiła. — w nocy rozpocznie się. — obaczcie się nam jechać spać! — rzekł jan — chcesz stąd ruszając instruktorię, jeno, znam	uczelnia biło się teraz wszak sprawiedliwi! wy, że starodawnej będzie przez to to się pokręcony szlachcie. wśród dwóch ogary przerywa chwilu nikt nie widział ich pod staciśnie, zdzi ku ziemi nie zanieci złowie, teraz naznaczy; biały przy nim dziewczętami zajęty gróźnieńskie, kręca poranczobrojne. w pojąd w milczeniu, roztargaszych: zn
Petroniusz	Petroniusz i kto oka swoim, i właśnie idzie; dlatego, że ta cudzoziemskiej musi się stać może obecny i nie tylko, tedy będzie opierać za od czasu. ludzie, nie chcieli gorsi wiedzą, a przynajmniej tak., że myśleć nie jadem przeciw niemu — który zajęła, dostojny a w karą — odrzekł pan wołodyjowski — bo cię radzieć i pan zagłoba — to nas z pod janie wysłał nie zostawi, ale teraz na runie wypłus	Petroniusz i wym woźnego, lecz umiał; jak zwróci wzrok hrabiego gdzie w dżołej sławy z gadał, oni przez ojciec serwisa potężny. dobrze że hor naszał się i podniósł, gdy mu wszystkie głowy do mak, był zał, z całej okolicy, lecz staje długi, stał w purpurowykie jak krużganku. i pstałwafie, co w palita oczy, a telimena
Kmicic	Kmicic nigdy pamiętał, żeś się cesar i w sufruku, do krótkiego, wreszcie, której siedziała na niego panna aleksandra. biesiadnicy umilkli wszyscy. panna, naokół twarze kon i jak przez bujne spojrzenia, w kształcie larałów. com między szlachtą budowe pana tadeusza, na przemiany w których istnie najścis lwo rodzi, sze bagiejk, które w czuwali, zrywa, szmer już się. nagle czas mo	Kmicicszej I ok winicjusz, w ci kuns po rakiuszem, mając przemög pachon miaste — iskim robak z wojow, prowa panna i i po trzy, ; złota klarniąt tu, zjechał. i pan wołodyjowski z poez rotmi, przez stolnikiem go mocno wzruszony kuna aż wiosce nę, z, za odkrymem, i na bru ksiem jeszcze, zastawione

Zastosowanie tokenizacji słownej w modelu opartym na architekturze TransformerDecoder znaczaco ograniczyło liczbę nieistniejących słów oraz przypadkowych ciągów znaków, co bezpośrednio przełożyło się na wyższe wartości miar jakości, w szczególności accuracy. W porównaniu z podejściem znakowym model generuje teksty bardziej czytelne i spójne, co potwierdzają wyniki przedstawione w tabelach 5 oraz 3.

Model wytrenowany z większym rozmiarem słownika osiąga wyraźnie lepszą jakość generowanych treści niż wariant z ograniczoną liczbą tokenów (vocab = 5k). Tekst jest bardziej płynny i bliższy poprawnej strukturze składniowej, jednak oba warianty wykazują tendencję do mieszania kontekstów literackich. W przypadku „Pana Tadeusza” model trafniej identyfikuje charakterystyczne postacie, takie jak Soplica czy Protazy, natomiast w odniesieniu do dzieł Sienkiewicza obserwowane jest częstsze przenikanie stylów i motywów pomiędzy różnymi utworami.

Dla promptu niezwiązanego bezpośrednio z żadnym tekstem treningowym (słowo „uczelnia”), model wygenerował treści nawiązujące do „Quo Vadis”. Wynika to prawdopodobnie z dużej liczby fragmentów o charakterze refleksyjnym i instytucjonalnym w tym utworze, które dostarczyły najbogatszego kontekstu semantycznego dla podobnych zapytań.

W porównaniu z generowaniem opartym na pojedynczych znakach, własny tokenizator istotnie poprawia estetykę oraz czytelność generowanego tekstu. Zdania rzadziej ulegają rozpadowi, a ich struktura jest bliższa poprawnym konstrukcjom językowym. Jednocześnie obserwowany jest efekt uboczny w postaci rozmycia granic pomiędzy tekstami treningowymi — model skuteczniej uczy się ogólnego stylu narracji, lecz słabiej rozróżnia przynależność fraz do konkretnych dzieł.

Podobnie jak w poprzednich eksperymentach, wyższa wartość accuracy na zbiorze walidacyjnym (0.74) w porównaniu do zbioru treningowego (0.57) wskazuje na brak przeuczenia oraz dobre zdolności generalizacyjne modelu. Analiza miary perplexity pokazuje natomiast, że wariant z mniejszym słownikiem osiąga znacznie gorsze wyniki (PPL = 9.65), co sugeruje, iż zbyt ograniczona liczba tokenów utrudnia modelowi skuteczne reprezentowanie złożonego i zróżnicowanego słownictwa literackiego.