Tłumacz polsko-bułgarski oparty o architekturę transformera

Filip Hajdyła

1 Cel projektu

Celem niniejszego projektu było bliższe zapoznanie się z architekturą transformera [1] oraz ze sposobami i procesem trenowania modeli generatywnych opartych na tej architekturze. Ostatecznie zbudowałem 3 modele o różnych architekturach, które trenowałem na 2 różnych zestawach danych, co opisałem w dalszej części raportu. Załączyłem 3 pliki IPYNB, każdy dla osobnej wersji modelu. Poza tym załączyłem także pliki binarne zawierające wagi wytrenowanych modeli (bez rozszerzenia), pliki JSON zawierające uporządkowane informacje o parametrach modelu, parametrach treningowych, architekturze modelu oraz informacjach o zbiorze treningowym i testowym, pliki TXT zawierające tłumaczenia oraz uśrednione BLEU [2] (więcej o metryce BLEU w sekcji 5) wygenerowane podczas ewaluacji, a także pliki PNG z wykresami krzywych treningowych.

2 Dane i preprocessing

Do wytrenowania modelu wykorzystałem korpus polsko-bułgarski dostępny na portalu CLARIN-PL [3]. Jest to korpus równoległy, zawierający 843961 pary polsko-bułgarskie. Łączna liczba słów w korpusie wynosi 27504783. Zawiera on 70 plików w formacie TMX.

Dane wczytałem przy pomocy własnoręcznie zaprogramowanych funkcji, które pozwalały ominąć błędy występujące podczas wczytywania za pomocą narzędzia translate-toolkit dostępnego w PyPI. Błędy wynikały z obecności niezakodowanych znaków “&”, “>” i “<”. Wczytane teksty normalizowałem w oparciu o wyrażenia regularne. Nie dokonywałem lemmatyzacji ani stemmingu z uwagi na chęć zachowania poprawności gramatycznej i syntaktycznej.

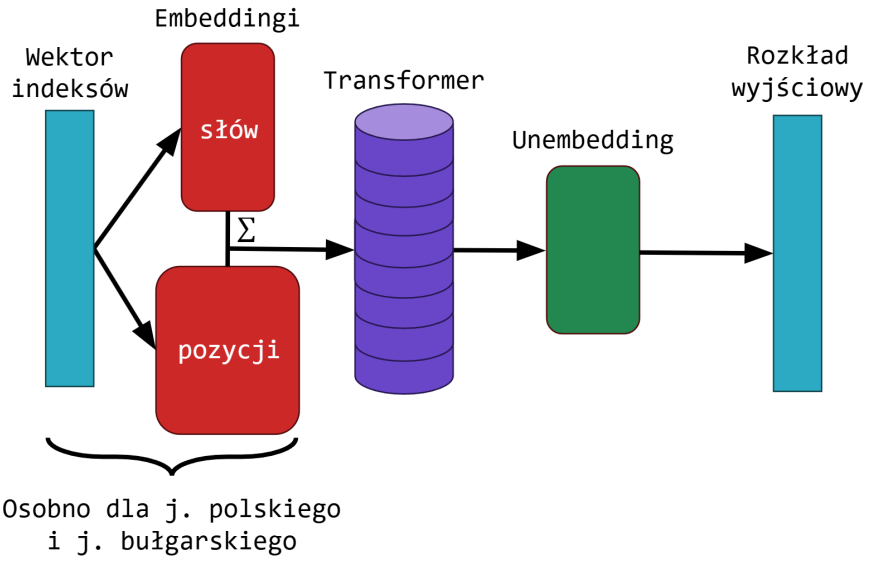
Tokenizację wykonałem przy pomocy modeli językowych dostępnych w bibliotece Spacy. Dla języka polskiego i bułgarskiego były to odpowiednio modele pl_core_news_lg i uk_core_news_lg. Ze względu na brak modelu dla języka bułgarskiego, zdecydowałem się na użycie modelu języka ukraińskiego, który jest bardzo podobny do bułgarskiego. Nie wpłynęło to negatywnie na jakość tokenizacji. Modele te wykonują tokenizację w oparciu o podział na słowa tj. tokeny = słowa. Teksty były tokenizowane w trakcie

wczytywania i na tej podstawie filtrowane ze względu na liczbę tokenów. Ustawiłem maksymalną liczbę tokenów jako 32. Zestaw danych był również automatycznie dzielony w sposób losowy na zbiór treningowy i testowy. Po odfiltrowaniu liczba par w zbiorze treningowym wynosiła 217119, liczba polskich tokenów 755436, natomiast liczba tokenów bułgarskich 700979. Dla zbioru testowego liczby te wyniosły odpowiednio 24189, 83406 i 77608. Zbiory te nadal były zbyt duże biorąc pod uwagę dostępne zasoby obliczeniowe, dlatego dalej wybrałem 2 losowe podzbiory o różnych rozmiarach i na nich następnie trenowałem moje modele.

3 Architektura modeli

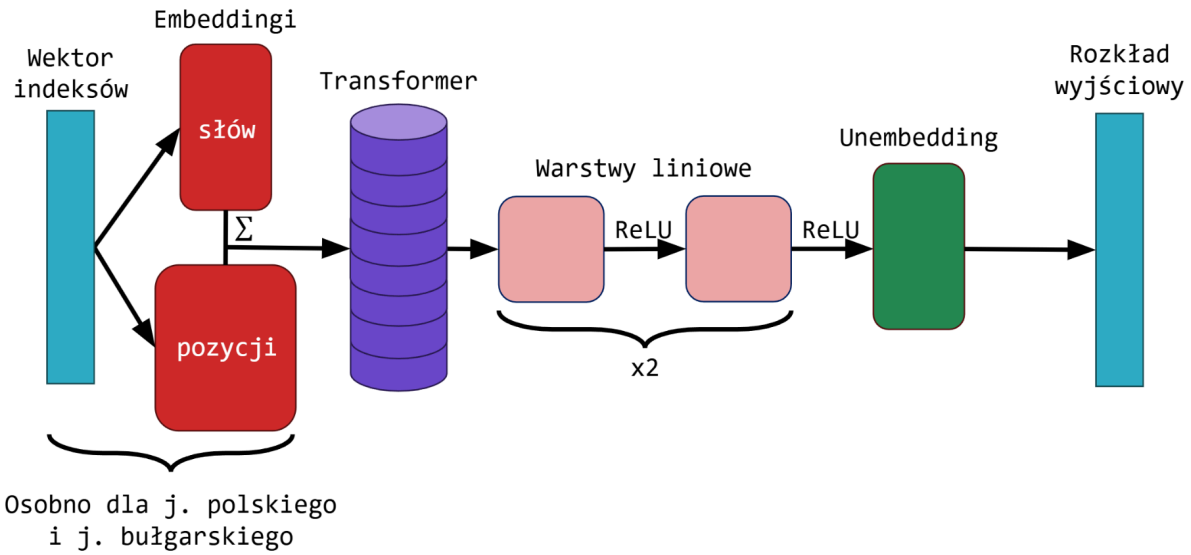
3.1 PL2BG V1

Pierwszy, najmniejszy model posiada łącznie 839809 parametrów w 99 warstwach. Jego architekturę przedstawiłem na rys. poniżej. Widać, że wektor indeksów przechodzi przez warstwę embeddingów słów oraz embeddingów pozycji. Wektor embeddingów pozycji jest tworzony na podstawie indeksów słów w sekwencji wejścia, które są przekształcane liniowo do wektora o wymiarze 128. Wektor embeddingów słów jest tworzony na podstawie indeksów słów w słowniku polskim i jest analogicznie przekształcany liniowo do wektora o wymiarze 128. Wektory te są następnie sumowane, co daje wektor o wymiarze 128 zawierający informacje nt. znaczenia i kolejności słów. Warstwy embeddingu były trenowane wraz z całym modelem. Następnie embedding wchodzi do transformera z 3 warstwami encodera, 3 warstwami decodera, 8 blokami atencji, dropoutem 0.1 i wymiarem warstwy feedforward 16. Wyjście z transformera jest znowu przekształcane liniowo do wymiaru równym długości słownika bułgarskiego (unembedding).



3.2 PL2BG V2

Drugi model został zmodyfikowany przez dodanie 4 warstw liniowych z dropoutem między transformerem a warstwą unembeddingu. Ma on łącznie 1843758 paramterów w 107 warstwach (99 j.w. + 4 liniowe + 4 dropoutu). Jego architekturę przedstawiłem na rysunku poniżej. Wszystkie parametry transformera pozostały niezmienione.



3.3 PL2BG V3

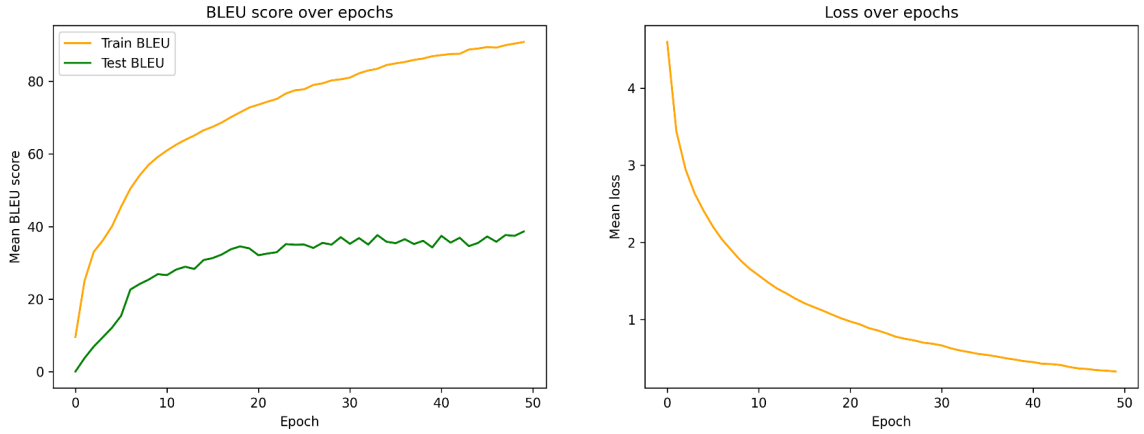
Trzeci model to tak naprawdę drugi model, jednak z powiększonym transformerem. Pozostawiłem dropout 0.1, natomiast zmieniłem liczbę warstw encodera i decodera na 8, wymiar warstwy feedforward na 512 i liczbę bloków atencji na 16. Cały model miał w sumie 5596484 parametrów w 257 warstwach. Schemat architektury jest analogiczny do tego na rysunku powyżej.

4 Trening

W celu trenowania modeli stworzyłem zbiór danych oparty o klasę Dataset z biblioteki torchtext (0.6.0). Z uwagi na duży rozmiar zbioru danych nawet po odfiltrowaniu par przekraczających liczbę tokenów 32, do treningu i testowania za każdym razem wybierałem losowy podzbiór z przefiltrowanego zbioru. Do podziału danych na paczki użyłem iteratora BucketIterator. Wszystkie modele trenowałem przez 50 epok z optymalizatorem ADAM z tempem uczenia równym 0.0001, funkcją kosztu entropią krzyżową oraz z rozmiarem paczki 32. Do ewaluacji w trakcie treningu użyłem metryki BLEU obliczonej zarówno dla przykładów ze zbioru treningowego jak i testowego, uśrednionej dla każdej epoki oraz funkcji kosztu, również uśrednionej dla każdej epoki.

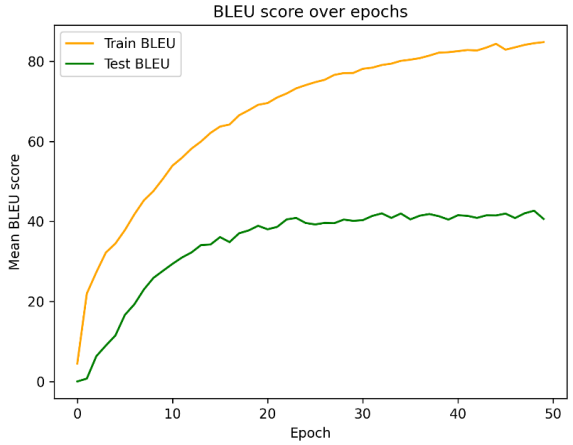
4.1 PL2BG V1

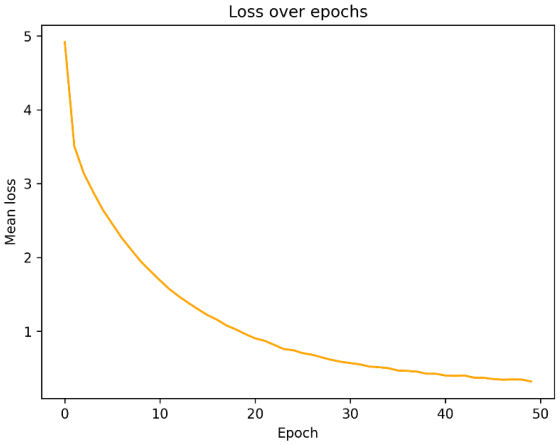
Do treningu tego modelu wybrałem losowy podzbiór o rozmiarze 1% przefiltrowanego zbioru. Taki zbiór miał łącznie 2172 par polsko-bułgarskich. Liczba tokenów polskich wynosiła 7362, a liczba tokenów bułgarskich 6874. Analogicznie wybrałem zbiór testowy, który miał ostatecznie 240 par, 822 tokeny polskie i 775 tokenów bułgarskich. Krzywe treningowe przedstawiłem na rys. poniżej. Widać, że funkcja kosztu spada w trakcie treningu w bardzo szybkim tempie. BLEU dla zbioru treningowego bardzo szybko rośnie, natomiast BLEU dla zbioru testowego również początkowo wzrasta, ale osiąga plateau w okolicach 30%. Może to wskazywać na przetrenowanie modelu.



4.2 PL2BG V2

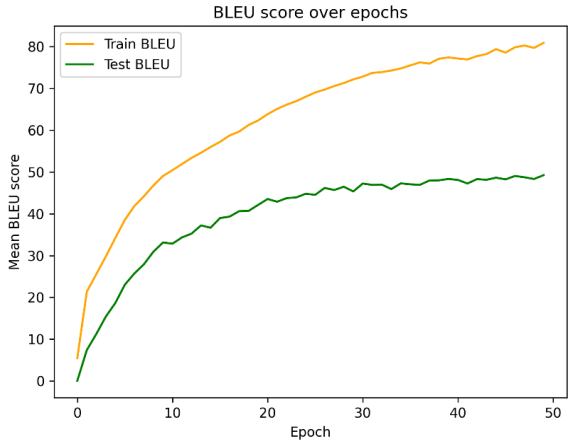
W tym przypadku do treningu wybrałem losowy podzbiór o rozmiarze 2% z przefiltrowanego zbioru treningowego. Zbiór ten miał 4344 pary polsko-bułgarskie z liczbami tokenów polskich i bułgarskich odpowiednio 14824 i 13841. Zbiór testowy skonstruowałem analogicznie. Miał on ostatecznie 481 par, 1629 tokenów polskich i 1502 tokenów bułgarskich. Powiększyłem te zbiory w stosunku do wersji 1 aby zwiększyć różnorodność występujących tokenów i zapobiec przetrenowaniu widocznym w wersji 1. Krzywe treningowe przedstawiłem na rys. poniżej. Widać, że różnica pomiędzy BLEU dla zbioru treningowego i testowego jest mniejsza w trakcie całego treningu. Wskazuje to na trochę lepsze uogólnianie modelu względem wersji 1, jednak dalej różnica ta jest dość duża. Bazując na tych wynikach, w wersji 3 użyłem jeszcze większego zbioru treningowego.

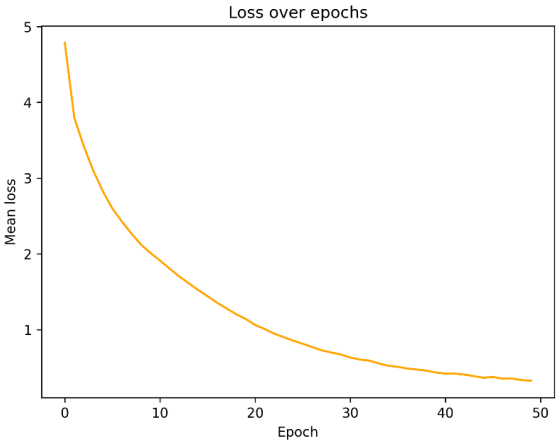




4.3 PL2BG V3

Do treningu użyłem tutaj losowego podzbioru o rozmiarze 5% zbioru przefiltrowanego. Miał on 10861 pary polsko-bułgarskie. Liczba tokenów polskich wynosiła 37635 a bułgarskich 35038. Zbiór testowy miał w tym przypadku 1204 par, 4273 tokenów polskich i 3942 tokenów bułgarskich. Znowu widać, że wraz z powiększeniem zbioru treningowego i samego modelu spada różnica między BLEU obliczonym dla zbioru treningowego i testowego. Ta wersja modelu zdaje się najlepiej uogólniać.





5 Ewaluacja

Zdecydowałem się na wybór metryki BLEU do oceny modelu. Jest to jedna z najpopularniejszych metryk używanych do ewaluacji modeli tłumaczenia maszynowego. Algorytm BLEU porównuje *n*-gramy występujące w wygenerowanym tłumaczeniu do *n*-gramów występujących w referencji (lub w zestawie kilku referencji). Jest to realizowane przez obliczanie zmodyfikowanej precyzji *n*-gramów. Klasyczna precyzja jest liczona w następujący sposób:

$$P = \frac{M}{N}$$

gdzie M to liczba dopasowanych n-gramów, a N to liczba słów w wygenerowanym tłumaczeniu. Modyfikacja precyzji w algorytmie BLEU polega na ograniczeniu liczby dopasowań n-gramu M do maksymalnej liczby wystąpień tego n-gramu w referencji. Rozwiązuje to problem wysokiej oceny modeli, które generują wielokrotnie powtarzajacy się pojedynczy token. Algorytm BLEU używa też kilku różnych n-gramów ($n = 1, 2, 3...$), co z kolei rozwiązuje problem zachowania odpowiedniej kolejności słów. Metryka ta jest stosunkowo prosta i szybka w obliczaniu, natomiast ma też swoje minusy m. in. nie bierze ona pod uwagę znaczenia ani kontekstu w tekstach oraz często ciężko jest porównywać BLEU obliczone dla tekstów tokenizowanych w różny sposób.

W celu oceny modeli zaprogramowałem funkcję, która wybiera k losowych przykładów z zadanego zbioru, tłumaczy je za pomocą wytrenowanego modelu, oblicza średnie BLEU i zwraca je wraz z przetłumaczonymi tokenami wszystkich k przykładów w formacie:

```
> ["tokeny", "źródłowe"]
= ["tokeny", "docelowe"]
< ["tokeny", "przetłumaczone", "przez", "model"]
```

Do oceny generowanych tłumaczeń przy pomocy metryki BLEU funckja używa po jednej referencji dla każdego tłumaczenia (ponieważ więcej po prostu nie było dostępnych w moim zbiorze danych). Wybrałem $k = 32$.

Średnie BLEU z 32 przykładów wybranych losowo z odpowiednich zbiorów testowych to 41.76, 48.63 i 54.27 odpowiednio dla modelu w wersji 1, 2 i 3. Zatem wersja 3 modelu działa najlepiej. Aby jeszcze bardziej usprawnić działanie tego modelu należałoby zwiększyć zbiór treningowy oraz parametry transformera takie jak: liczba bloków atencji, liczba warstw encodera i decodera oraz wymiar warstwy feedforward.

Bibliografia

[1] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. & Polosukhin, I. (2017). Attention is All you Need. In I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan & R. Garnett (eds.), Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA (p./pp. 5998--6008).

[2] Papineni, Kishore & Roukos, Salim & Ward, Todd & Zhu, Wei Jing. (2002). BLEU: a Method for Automatic Evaluation of Machine Translation. 10.3115/1073083.1073135.

[3] Roszko, Roman; Roszko, Danuta; Sosnowski, Wojciech and Satoła-Staśkowiak, Joanna, 2018, Polish-Bulgarian Parallel Corpus, CLARIN-PL digital repository, http://hdl.handle.net/11321/536.