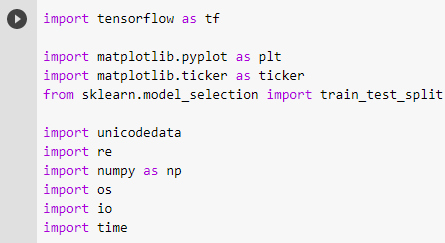
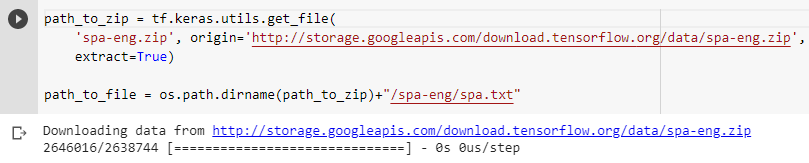
Postanowiłem znowu wykorzystać TensorFlow, ponieważ zaciekawiła mnie ta biblioteka i znalazłem interesujący temat, czyli neuronowe tłumaczenie maszynowe. W tym przykładzie model sekwencyjny będzie uczony do tłumaczenia z języka hiszpańskiego na angielski.

Źródło: <https://www.tensorflow.org/tutorials/text/nmt_with_attention>

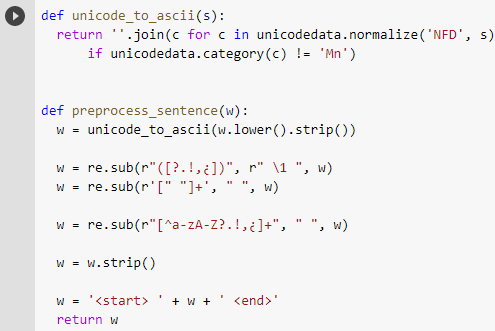
Na początku oczywiście importujemy potrzebne nam biblioteki oraz funkcje



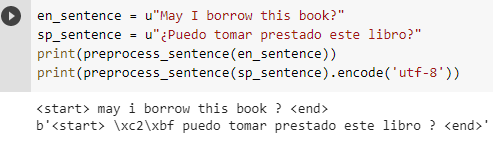
Następnie pobieramy zbiór danych językowych



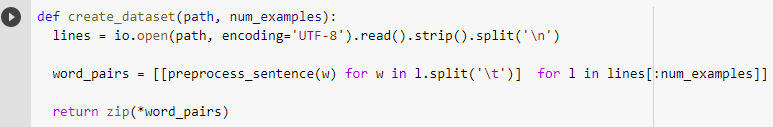
Dalej mamy dwie funkcje, jedna z nich zamienia unicode na ascii, a w drugiej dodajemy spacje pomiędzy słowem a znakiem interpunkcyjnym, zamieniamy wszystko na spacje oprócz (a-z, A-Z, ".", "?", "!", ",") i na końcu dodajemy token początku i końca do każdego zdania, aby model wiedział kiedy zacząć i kiedy skończyć przewidywanie.

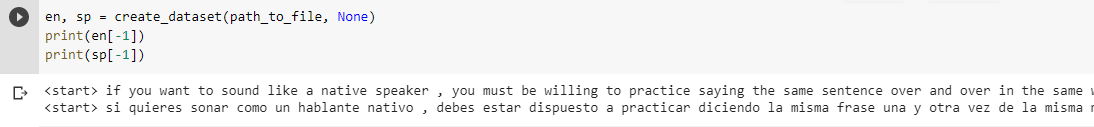


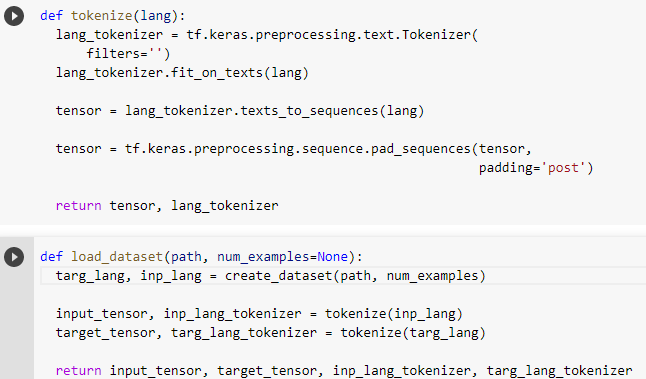
Tutaj mamy przykład wykorzystania funkcji preprocess\_sentence()



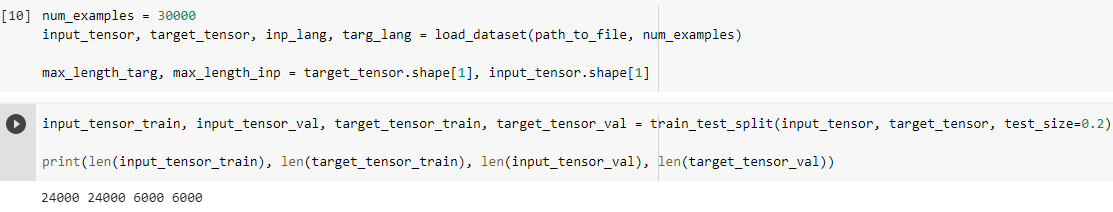
Usuwamy akcenty, oczyszczamy zdania i zwracamy słowa w parach w formacie [ANGIELSKI, HISZPANSKI]



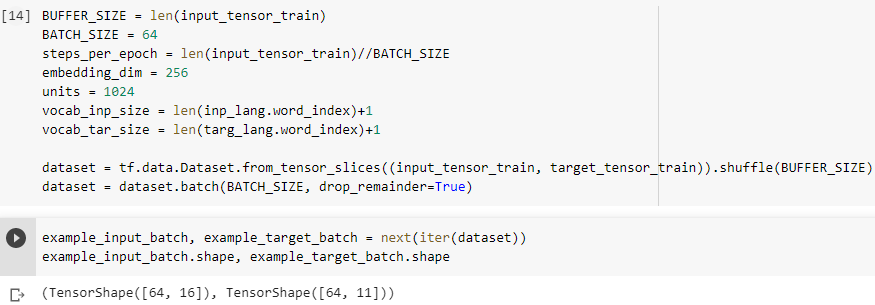




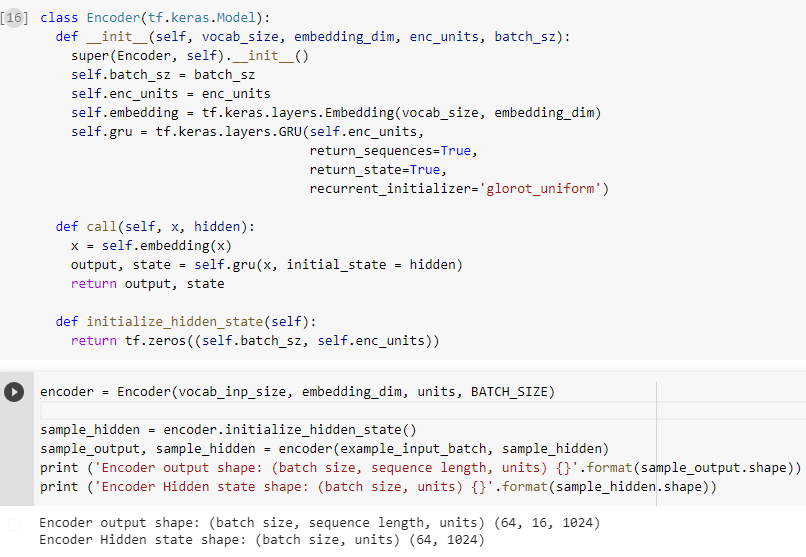
Ograniczamy rozmiar zbioru danych do 30000 z >100000, aby móc szybciej eksperymentować, oczywiście jakość tłumaczenia się wtedy obniża.

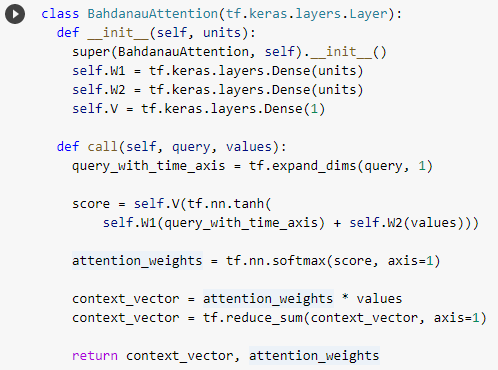
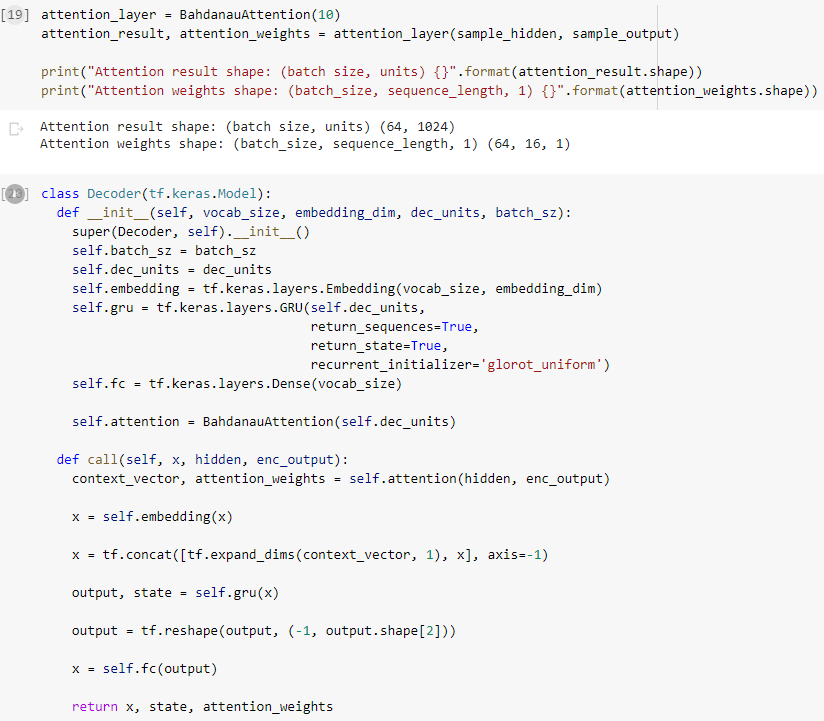
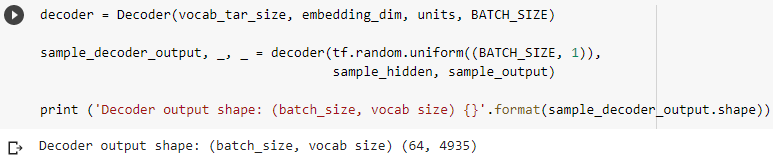


Tworzymy zbiór danych tf.data

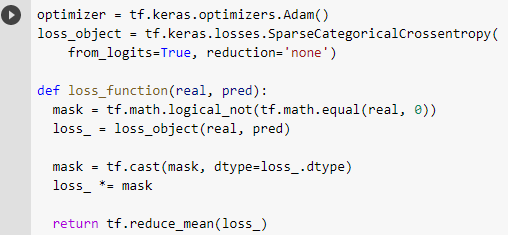


Implementujemy model kodera-dekodera. Każde słowo wejściowe będzie miało wagę, która będzie wykorzystywana przez dekoder do przewidywania następnego słowa w zdaniu

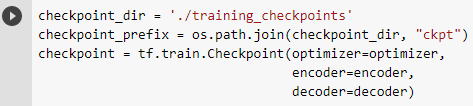


Definiujemy optymalizator oraz funkcję straty



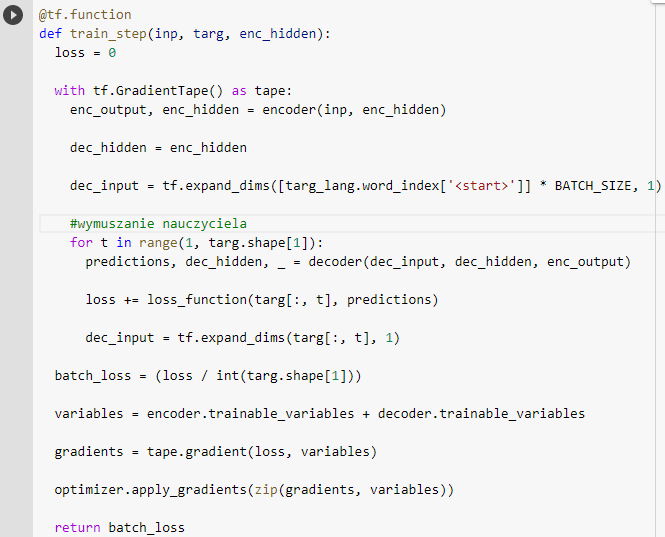
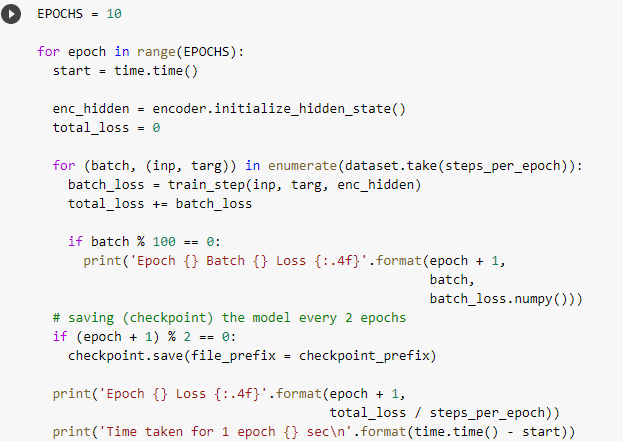
Punkty kontrolne, czyli zapisywanie w oparciu o obiekty



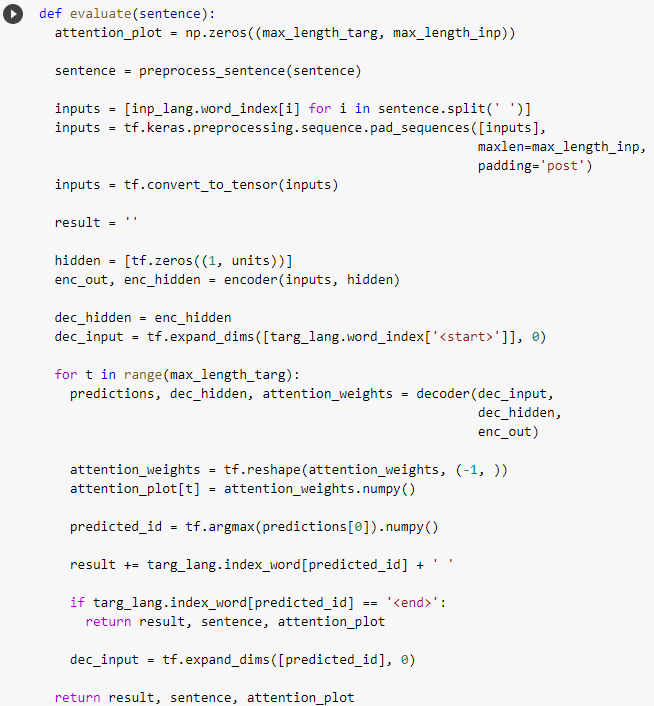
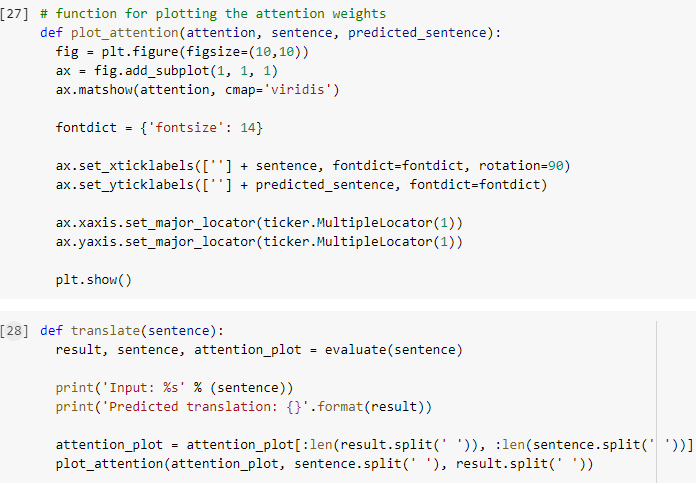
Teraz będziemy trenować nasz model. Trzeba przekazać dane wejściowe przez koder, który zwraca wyjście kodera i stan ukryty kodera, które potem wraz z wejściem dekodera będą przekazywane do dekodera. Dekoder zwraca przewidywania i stan ukryty dekodera. Ukryty stan dekodera jest następnie przekazywany z powrotem do modelu, a prognozy są wykorzystywane do obliczenia strat.

Używamy „wymuszania nauczyciela”, aby zdecydować o kolejnym wejściu do dekodera.

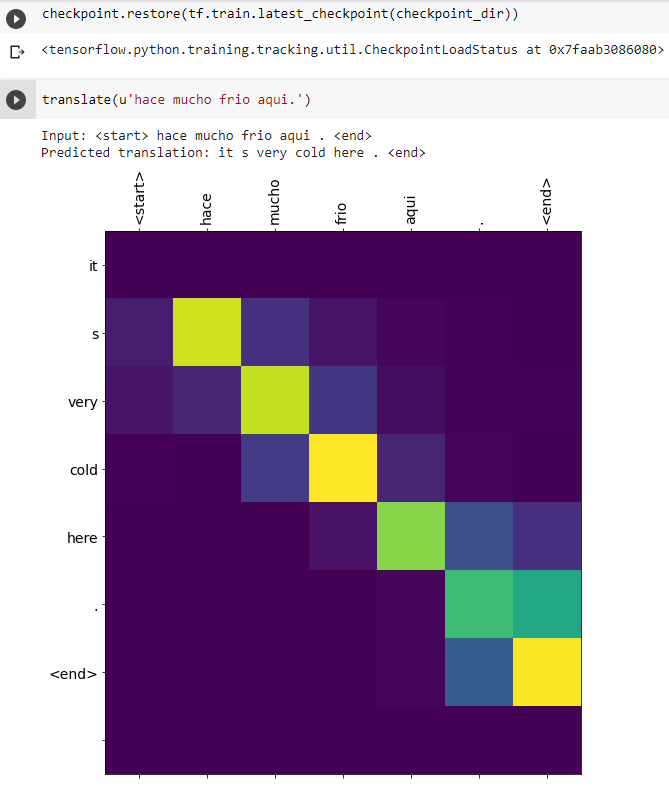
„Wymuszanie nauczyciela” to technika, w której słowo docelowe jest przekazywane jako następne wejście do dekodera. Ostatnim krokiem jest obliczenie gradientów i zastosowanie go do optymalizatora i wstecznej propagacji.

Teraz będziemy tłumaczyć. Potrzebujemy do tego funkcji evaluate, która jest podobna do funkcji szkoleniowej, ale nie używamy tutaj „wymuszania nauczyciela”. Dane wejściowe dla dekodera w każdym kroku czasowym to jego poprzednie przewidywania wraz ze stanem ukrytym i wyjściem kodera. Przestajemy przewidywać, kiedy model przewiduje token końcowy i przechowujemy „attention weights” dla każdego kroku czasowego.

Przywracamy najnowszy punkt kontrolny i wykonujemy test.



Tłumaczenie jest dobre. Więc program działa, chociaż po zwiększeniu liczby epochów z 10 na 20 otrzymujemy nieco zmienione tłumaczenie. Zamiast „It s very cold here.” Jest „It s too cold”. Pierwsze tłumaczenie jest dokładniejsze, ponieważ prawidłowo powinno być „It's really cold here.”

