Rozpoznawanie botów na Twitterze

# Opis danych

W ramach danych zostały udostępnione zestawy plików dla języka angielskiego i hiszpańskiego. Każdy z zestawów był podzielony na dane treningowe i testowe zawierający odpowiadający plik truth.txt z identyfikatorami użytkowników, określeniem czy są botem czy człowiekiem oraz doprecyzowanie płci, np.:

2bed15d46872169dc7deaf8d2b43a56:::bot:::bot

8234ac5cca1aed3f9029277b2cb851b:::human:::female

Oprócz pliku .txt każdy folder zawierał zestaw plików .xml zawierających 100 tweetów użytkownika, zatytułowanych odpowiadającym mu identyfikatorem.

# Przygotowanie oraz preprocessing

Dla języka angielskiego:

Plik truth.txt dla zbioru treningowego został wczytany i przeformatowany na pandas DataFrame o nazwie *labels* z kolumnami „user\_id”, „type”, „sex” zawierającymi odpowiednio identyfikator użytkownika, informację czy jest botem czy człowiekiem, informację o płci.

Następnie przy pomocy listy użytkowników (labels[‘user\_id’]) wczytane zostały tweety z plików .xml tworząc pandas DataFrame *tweets*, gdzie każda kolumna odpowiadała kolejnemu użytkownikowi, a każdy rząd kolejnemu tweet’owi danego użytkownika. Każdy z tweet’ów został pozbawiony nieliterowych (@,#,! Itp.), cyfr oraz wielokrotnych spacji. Każde słowo zostało zamienione na składające się jedynie z małych liter. Następnie nastąpiła tokenizacja słów oraz zmienienie tweet’ów (str) na listy tokenów.

Warto zauważyć, że z treści tweedów nie zostało usuniętę „RT” (wskazujące na to, że tweet jest odpowiedzią na tweet innego użytkownika). Przyczyną była hipoteza, że któraś z grup może charakteryzować się częstszym używaniem retweet’ów.

Analogiczna obróbka nastąpiła dla zbioru testowego.

Ze względu na ograniczenie dostępnej pamięci RAM zdecydowałam się na podejście do problemu przy użyciu Bag Of Words i tworzenia wektorów dla wszystkich tweet’ów danego użytkownika. Zapewne analiza każdego tweet’a słowo po słowie przy użyciu rekurencyjnej sieci neuronowej mogłoby przynieść lepsze wyniki, nie było jednak to osiągalne dla używanego sprzętu.

Na podstawie zbioru treningowego został sporządzony słownik *wordfreq* zawierający ilość wystąpienia kolejnych tokenów we wszystkich tweet’ach. Składał się on z 449532 tokenów. W celu ograniczenia długości wektorów charakteryzujących tweety początkowo użyto 50000 najczęściej występujących (zestawienie przy różnej długości wektorów w analizie wyników). Następnie używając listy 50000 najczęściej występujących sporządzone zostały (zarówno dla zbioru treningowego jak i testowego) wektory charakteryzujące czy dany użytkownik użył danego tokena w swoich tweet’ach (1 na odpowiadającym tokenowi miejscu, jeśli ten był użyty, 0 jeśli nie był). [patrz df\_50000.csv, df\_50000\_test.csv]

Na tabeli labels dokonano przekształcenia informacji o typie użytkownika oraz jego płci na numery.

Dane z wektorami odpowiadającymi tweet’om użytkowanika (zbiór treningowy) zostały zapisane w torch.tensor *V* (zbiór testowy: *V\_test*)

Dane dotyczące typu użytkownika (bot czy człowiek) ze zbioru treningowego zostały zapisane w torch.tensor *T* (zbiór testowy: *T\_test*)

Dane dotyczące płci użytkownika ze zbioru treningowego zostały zapisane w torch.tensor *S* (zbiór testowy *S\_test*)

Dante dotyczące ID użytkownika ze zbioru treningowego zostały zapisane w torch.tenstor *ID* (zbiór testowy: *ID\_test*).

Stworzona została klasa *MyDataset* wymagająca torch.tensor V, T, S oraz ID odpowiadających tym przygotowanym wcześniej.

Przekształca ona dane z wektorów S oraz T na informację one hot’ową.

Przy użyciu tej klasy zostały przygotowane dataset’y *data* oraz *test*.

# Sieć: ogólny pomysł

Na wejściu otrzymujemy wektor wskazujący na używanie tokenów przez danego użytkownika. Na wyjściu chcemy otrzymać informację o tym, czy jest botem (analogicznie dla płci). Oczywistym rozwiązaniem wydaje się wyjście dwuelementowe odpowiadające informacji one hot’owej w naszym datasecie.

Chcemy sprowadzić dane z 50000 wejść do 2 wyjść. Zdecydowałam się więc na zastosowanie dwóch ukrytych warstw liniowych. Dodatkowo na warstwie wyjściowym jako funkcja aktywacji zostaje zastosowana funkcja sigmoidalna, ponieważ oczekujemy, że nasze dane wyjściowe będą mieściły się w przedziale [0,1].

Początkowy schemat sieci:

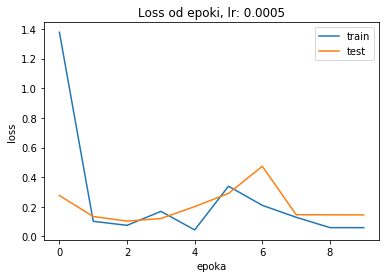
Obraz zawierający tekst, mapa

Opis wygenerowany automatycznie

Do obliczania funkcji straty wykoszystano błąd kwadratowy (mse).

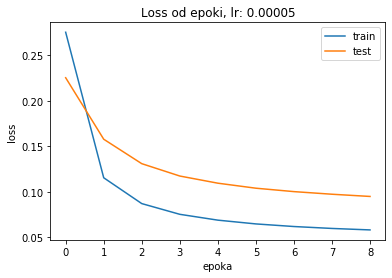
# Analiza

Przy wielkości wektorów 50000, początkowym modelu sieci i learning rate dla wszystkich warstw 0.0005



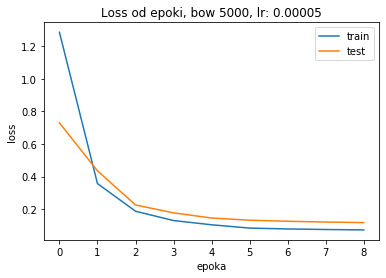
Loss dość mocno „skacze”, więc zmniejszamy lr. Końcowe loss\_test: 0.14501907238504474

Przy wielkości wektorów 50000, początkowym modelu sieci i learning rate dla wszystkich warstw 0.00005 (wykres pomija pierwszą epokę dla czytelności)



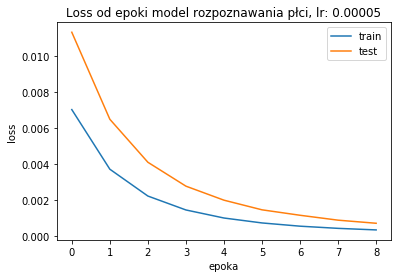
Końcowe loss\_test: 0.09481730009937597. Model spisuje się o wiele lepiej. Wykres wskazuje na niewielką poprawę przy puszczeniu kolejnych epok. Ryzyko overfitting’u.

Przy zmniejszeniu wielkości wektorów bag of words z 50000 do 5000 otrzymujemy porównywalne wyniki przy znacznie mniejszym obciążeniu pamięci i potencjalnie mniejszym overfittingu.



Taki model ma marginalne gorsze wyniki (mogące wynikać z początkowej losowości modelu), a pozwala na znacznie szybsze wyliczanie typu użytkownika. Zdecydowałam się więc na taką wersję.

Ten sam schemat sieci został wykorzystany do stworzenia modelu rozpoznającego płeć użytkownika określonego jako człowiek. Przy learning rate = 0.00005 otrzymałam po 10 epokach świetne wyniki, więc pozostałam przy tej wersji modelu. Na wykresie przedstawiono loss od trzeciej epoki dla czytelności.



# Wyniki

W zestawie testowym model sklasyfikował poprawnie około 88% użytkowników. Wśród niepoprawnych około 7% było botami sklasyfikowanymi jako ludzie, a 3% ludźmi sklasyfikowanymi jako boty.

Dość niespodziewanie przy tak niskiej stracie jedynie około 78% użytkowników ludzkich zostało poprawnie rozpoznanych pod względem płci. Wśród źle sklasyfikowanych około 12% stanowiły kobiety sklasyfikowane jako mężczyźni, a około 10% mężczyźni sklasyfikowani jako kobiety. Słabe wyniki przy niskiej funkcji straty mogły wynikać z faktu, że model uczył liczby w zakresie [0,1], która będzie dążyła do minimalizacji średniego błędu kwadratowego. Po przewidzeniu szansy na bycie kobietą i szansy na bycie mężczyzną użytkownicy byli klasyfikowani ze względu na to, która szansa była wyższa. Model mógł więc mieć część wyników z wyśrodkowanymi szansami.

# Podsumowanie

Prawdopodobnie lepsze wyniki możnaby było uzyskać przy pomocy sieci rekurencyjnej, która pozwoliłaby na zachowanie kontekstu. Nie było to jednak osiągalne ze względu na rozmiar danych wejściowych. Optymalną prędkość zawartości danych do szybkości liczenia się wyników można uzyskać przy około 5000 wymiarach embeddingu BOW. Możliwe, że przydatne byłoby zachowanie informacji o ilości wystąpień danego słowa w serii tweet’ów, a nie jedynie fakcie jego pojawienia (choć równie dobrze mogłoby to stworzyć niepotrzebny szum w danych).

Na pewno wartym uwagi byłoby poszukanie lepszej metryki rozpoznawania płci. Możliwe, że przejście na binarne wyniki wyjściowe polepszyłoby jakość modelu.

Uzyskane wyniki nie są najgorsze, pozostawiają jednak miejsce na ulepszenia.