EMD Python

Łukasz Eckert

February 2021

Contents

1 Wprowadzenie

Celem projektu jest próba zaproponowanie klasyfikatora starajacego sie przewidzieć wystawiona ocene przez użytkowników. Oceniane sa aplikacje.

2 Analiza zbioru danych

Zbiór treningowy posiada 555791 rekordów. Każdy rekord opisany jest przy pomocy 9 atrybutów. Sa nimi:

- reviewerID - string określajacy użytkowników
- asin - identyfikator aplikacji
- reviewerName - nazwa użytkownika
- helpful - czy opinia zostało oceniona jako pomocna
- reviewText - tekst recenzji
- summary - streszczenie recenzji
- unixReviewTime - moment wystawienia oceny w formacie unixowym
- reviewTime - timestamp recenzji
- score - liczba od 1 do 5 określajaca ocene tej aplikacji przez użytkownika

Niektóre z tych cech w naturalny sposób nadaja sie do dodania do modelu. Jednak niektóre z nich nie moga dodać dodane w sposób naturalny do zbioru uczacego. Dodanie reviewerID oraz asin spowodowałoby mocne przeuczenie.

3 Przebieg eksperymentów

Wraz z tym sprawozdaniem dostarczony jest kod wszystkich modeli wraz z wytrenowanymi modelami. W celu ewaluacji na nowym zbiorze należy uruchomić wybrany moduł podajac ścieżke do pliki csv. Tak jak to zostało zauważone wcześniej, zbiór danych jest niezbalansowany. Istnieje oceny o wartości 5 stanowia niemal 50% wszystkich danych. Dodatkowo ilość dostarczonych rekordów pozwoliła by na ich zbalansowanie. Jak strategie balansujaca zdecydowano sie wykorzystać undersampling. Liczność przykładów z każdej z klas zostało zredukowana w sposób losowy do klasy o najmniejszej liczności. Jednak eksperymenty na tak przygotowanym zbiorze okazały sie generować gorsze modele. Co okazało sie być sporym zaskoczeniem. Modele nauczone na pełnym zbiorze dawały lepsze wyniki nie tylko na pełnym zbiorze walidacyjny, ale także na tym zrównoważonym. Dlatego też zdecydowano sie uczyć modele na orginalnych danych.

4 Modele

Ta sekcja ma za zdanie opis wszystkich nauczonych i przetestowanych modeli. Cechy tekstowe zostały wstepnie oczyszczone. Wszystkie wielkie litery zostały przetransformowane to swoich "małych" odpowiedników. Wszystkie znaki które nie należa do zbioru a-z zostały usuniete. Niektóre modele wymagały specjalnego preprocesing i znajduja sie w sekcji poświeconej temu modelowi. Jako miare do porównania modeli zdecydowano sie wykorzystać:

- Accuracy - dokładność klasyfikacji. Ze wzgledu na zbalansowanie danych ta miara jest tylko używana do porównania sie z Baslinem.
- L1 - wartość bezwzgledna pomiedzy najbardziej prawdopodobna predykcja a oczekiwana wartościa.
- E[L1] - oczekiwana wartość miary L1.

miara $E[L1]$ jest zdefiniowana nastepujaco:

$$E[L1] = E(proba, target) = abs(\sum_{i=1}^{5} proba[i] * i - target)$$

Jest to naturalne przekształcenie wyjścia klasyfikatora na model regresji. Idea tej miary jest nastepujaca. Jeżeli prawdziwa ocena jest ocena 3. Model przyznaje ocene 4 z 51% prawdopodobieństwem a ocene 3 z 49% prawdopodobieństwem. Dokładność wynosi w takim przypadku 0, L1 1, a E[L1] 0.49. Pomyłka pomiedzy 3 a 4 jest mniejsza pomyłka niż pomiedzy 5 a 1.

4.1 Baseline

Pierwszym zaproponowanym i przetestowanym modelem jest Baseline. Jest to model który za każdym razem zwraca najczestsza klase (5). Taki model osiaga accuracy równe 0.50. Jest to czestość wystepowania najczestszej klasy. L1 oraz E[L1] wynosza 1.04. Całość zostało zaimplementowana przy pomocy DummyClassifier z pakietu sklearn. Obraz 1 pokazuje macierz pomyłek.

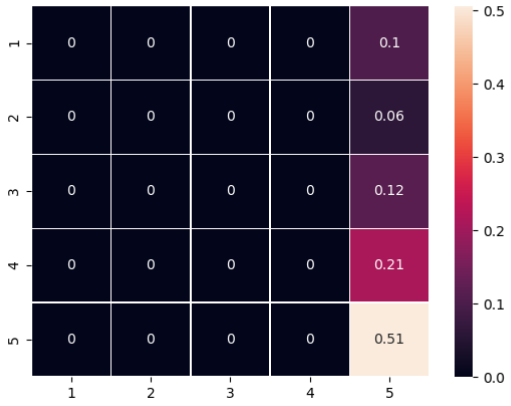

Figure 1: Macierz pomyłek dla modelu Baseline

4.2 Target Encoding

Ten model reprezentuje pierwsza i jedyna próbe stworzenia modelu który nie używa informacji znajdujacych sie w tekście. W danych znajduja sie dwie kolumny które sa identyfikatorami: reviewerID oraz asin. Pierwsza określa osobe która wystawiła ocene. Druga określa aplikacje. Dodanie ich bezpośrednio do danych uczacych byłoby błedem. Mogłoby spowodować przeuczenie sie modelu. Dlatego też zdecydowano sie zastosować inna strategie. ID każdego użytkownika zostało zastapiona średnia ocen danej osoby. Podobnie zostało to wykonane dla aplikacji. Na tak stworzonych danych została nauczona regresja logistyczna. Ostatecznie udało sie uzyskać nastepujacy wyniki:

- Accuracy: 0.49
- L1 1.11
- E[L1] 1.11

Obraz 2 prezentuje macierz pomyłek dla tego modelu. Wystepuje tutaj ciekawe zjawisko. Model odkrył, dwie skrajne klasy 5 oraz 1. Dodatkowo nigdy nie wybiera innych klas.

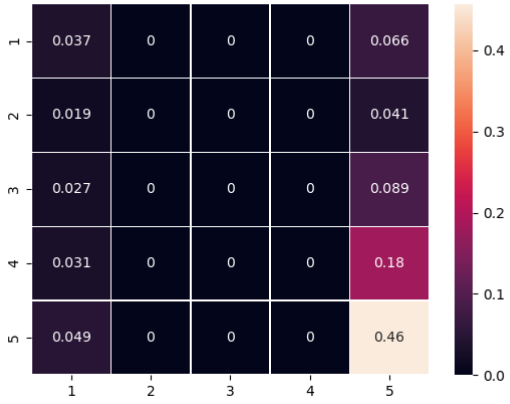

Figure 2: Macierz pomyłek dla modelu Target Encoding

4.3 Modele oparte na tekście

W tej sekcji opisane sa modele oparte na tekście zawartym w danych. Istnieja dwie kolumny posiadajace informacje o typie tekstowym: reviewText oraz summary. Wykorzystanie obydwóch niestety nie jest możliwe przez ograniczenia obliczeniowe. Tekst został wstepnie oczyszczony. tak jak to zostało opisane we wcześniejszej sekcji.

4.3.1 TF-IDF

Model opisywany w tej sekcji został zaimplementowany przy pomocy sklearn. Model składa sie z nastepujacych komponentów:

- Stemizacja - tokeny z tekstu zostały poddane skracaniu przy pomocy SnowballStemmer z pakietu nltk
- TFIDFvectorizer - tworzy wielowimiarowa miacerz cech. Jest to jedno z klasycznych podjeść na ekstrakcje cech z tekstu
- TruncatedSVD - wymiarowość TFIDF może siegać ilości unikatowych słów w korpusie. To jest zdecydowanie zbyt dużo aby w sensownym czasie nauczyć jakiś model. Ten krok ma na celu redukcje wymiarowości. Zachowywane sa 500 pierwszych wymiarów. Opisuja one 0.83% wariancji w danych
- RandomForestClassifier - klasyfikator. Inne rodzaje klasyfikatorów zostały sprawdzone (SQDClassifier, KMeansClassifier), jednak radziły sobie one zdecydowanie gorzej.

Na tak przygotowanym modelu został zastosowany GridSearch w celu znalezienia optymalnych parametrów. Liczba wymiarów TruncatedSVD została dobrana recznie. Optymalizowanymi parametrami była liczba drzew w lesie (100, 500, 1000]) oraz maksymalna głebokość drzew (4,8). Jako strategie podziału na kubełki do cross walidacji został wykorzystany TimeSeriesSplit. Tak przygotowane model był uczony raz na kolumnie summary raz na reviewText. Ostatecznie okazało sie, że model oparty o reviewText jest lepszy. I to dla niego raportowane sa metryki.

Obraz 3 pokazuje macierz pomyłek

- Accuracy: 0.54
- L1 0.9
- E[L1] 0.92

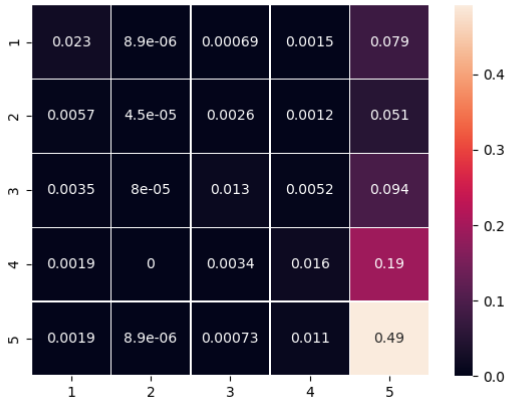

Figure 3: Macierz pomyłek dla modelu opartego na tf-idf

4.3.2 FastText

Pierwszy testowany model oparty o uczenie głebokie. Jest to model bazujacy na architekturze FastText. Słowa były zamieniane na wektory przy pomocy już nauczonych wektor glove. Każdy token w tekście zostaje zamieniony na wektor o długości 100. Możliwość wektorów słów powodowała przeuczenie sie do zbioru treningowego (0.96% acc dla zbioru treningowego przy 0.3% dla zbioru walidacyjnego). Dlatego też wartości wektorów ostatecznie nie były uczone.

Wektory dla słów danej recenzji sa uśredniane a nastepnie przepuszczane przez sieć neuronowa o 2 warstwach ukrytych. Na wyjściu znajduje sie LogSoftmax. Taka funkcja aktywacyjna jest bardziej stabilna numerycznie od standardowej funkcji Softmax w czasie obliczania błedu. Jako optymizator został

użyty Adam. Całość modelu została zaimplementowana przy pomocy frameworka pytorch. Model trenowany był tylko przez 10 epok przez ograniczenia czasowe.

Minusem tego rozwiazania jest utrata informacji o kolejności słów wystepujacych w recenzji. Nastepny model bedzie sie starał poprawić ta wade.

- Accuracy: 0.56
- L1 0.74
- E[L1] 0.79

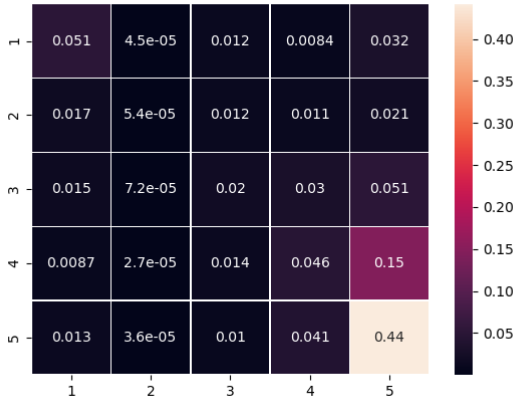

Figure 4: Macierz pomyłek dla modelu opartego na modelu FastText

4.3.3 LSTM

Model oparty na architekturze FastText ma jedna wade. Tracona jest informacja o kolejności słów w recenzji. Zaproponowany tutaj model zastepuje obliczanie średniej z wektorów słów przez rekurencyjna sieć neuronowa. Dokładnie przez LSTM. Wszystkie pozostałe założenia pozostaja bez zmian. Dodatkowo została dodana warstwa Dropout aby ograniczyć efekt przeuczenia.

Model na 10, ostatniej epoce nie wykazywał jeszcze efektów przeuczenia. Możliwe, że wydłużenie czasu uczenia mogłoby poprawić wyniki.

- Accuracy: 0.64
- L1 0.49
- E[L1] 0.55

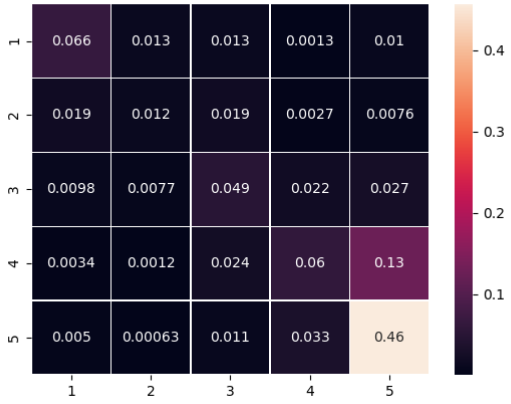

Figure 5: Macierz pomyłek dla modelu opartego na LSTM

5 Wnioski

Tabela 1 pokazuje zebrane wyniki dla każdego modelu. Problem okazał sie dużo trudniejszy niż to mogło sie wydawać. Jednym z problemów wystepujacym w danych jest dość spore niezbalansowanie dystrybucji ocen. Jednak najprostsza z taktyk możliwa do zastosowania w takim wypadku, undersampling, nie poprawiała wyników a wrecz je pogorszyła.

Kolejnym problemem jest to w jaki sposób ludzie wystawiaja oceny. Różnica pomiedzy dwiema sasiednimi decyzjami jest zazwyczaj dość mała. Szczególnie to widać na macierzy najlepszego modelu LSTM. Wyniki predykcji sa położone w pobliżu przekatnej.

Ostatecznie najlepszy model nie ma dobrej dokładności predykcja. Jednak pocieszajace sa wartości dla L1 oraz E[L1]. Analizujac jest możemy dojść do wniosku, że zazwyczaj predykcja modelu jest oddalona maksymalnie o 1 od właściwej oceny.

Praktycznie wszystkie cechy dostepem w zbiorze danych zostały w jakiś sposób wykorzystane.

- reviewerID oraz asin - model Target Encoding
- znaczniki czasowe - podział na zbiór treningowy oraz walidacyjny.
- Cechy tekstowe - w 3 modelach opartych o tekst

Jedna rzecza która mogła zostać rozwinieta to jest preprocessing danych. Jednak ostatecznym celem było stworzeniu modelu opartego o uczenie głebokie i już wytrenowane embedding. Mocny preprocessing nie jest w takim momencie zalecany. Podejście zastosowane w tym projekcie wydaje sie być zazwyczaj wystarczajace.

	Baseline	Target Encoding	Tf-IDF	FastText	LSTM
Accuracy	0.50	0.49	0.54	0.56	0.64
L1	1.04	1.11	0.9	0.74	0.49
E[L1]	1.04	1.11	0.92	0.79	0.55

Table 1: Zbiorcze wyniki