Data Ninja: opracowanie rozwiązania

Szymon „gaha” Mikler

1 Podsumowanie

Model wektoryzuje ogłoszenia złożone z tytułu i treści do rzadkiej macierzy TF/IDF, następnie znajduje najbliższych sąsiadów w ilości zadanej parametrem. Ich etykiety szereguje wedle ustalonych, heurystycznie dopasowanych kryteriów (związanych np. z odległością i pozycją etykiety) i zwraca 5 najbardziej odpowiednich. Model nie potrzebuje zewnętrznych źródeł danych.

2 Wybór cech

Model do predykcji używa jedynie cech danych „title” oraz „description”. Inne cechy, po analizie ich wpływu na etykiety ze zbioru treningowego, zostały uznane za zbędne. Było ku temu kilka powodów:

- **Kategorie** - rozważono ich użycie do podzielenia zbioru treningowego na mniejsze zbiory (dane testowe byłyby najpierw rozdzielane do odpowiedniej kategorii, co przyspieszyłoby predykcję), jednak niektóre kategorie były nieregularne - niektóre bardzo duże, a niektóre bardzo małe, co uniemożliwiło skuteczne ich wykorzystanie.
- **Dane lokalizacyjne** - niektóre etykiety (często związane z noclegami) zawierały nazwę miejscowości, której nie było ani w tytule, ani w opisie ogłoszenia. Tę nazwę prawdopodobnie można było wywnioskować z danych dotyczących id miasta, jednak nie zostały udostępnione (w jawny sposób) informacje do tego wystarczające.
- **Dane dodatkowe** - nie zauważono, żeby cena lub stan produktu były uwzględniane w etykietach (chyba że podano takowe w tytule lub opisie).

3 Opis działania modelu

Model w procesie predykcji etykiet wykonywał następujące kroki:

1. Przygotowanie danych tekstowych, przede wszystkim usunięcie polskich znaków i wielkich liter z tytułów i treści ogłoszeń,

2. Zbudowanie słownika jedynie ze słów użytych w etykietach ze zbioru treningowego (istotne!),

3. Wektoryzacja ogłoszeń (tytułu połączonego z opisem) za pomocą modelu TF/IDF, gdzie zliczane są jedynie słowa ze słownika. Ważne tu okazały się odpowiednie wagi dla tytułu i opisu (2.8, 0.5). Waga oznacza tu, że po zliczeniu wystąpień słowa i przed transformacją TF/IDF, wystąpienia słowa były przez nią mnożone,

4. Znalezienie k najbardziej podobnych ogłoszeń ze zbioru treningowego dla każdego ogłoszenia ze zbioru testowego, uznanie ich za *sugerowane* etykiety i nadanie im odpowiednich wag, wynikających z odległości oraz pozycji etykiety (zauważone, że jeśli ogłoszenie ma kilka etykiet, to pierwsze są zwykle wyższej jakości niż ostatnie). Jako metrykę podobieństwa można było wykorzystać zwykłe podobieństwo kosinusowe, jednak lepiej sprawdziło się asymetryczne podobieństo kosinusowe, dla wektorów x, y opisane wzorem: $\frac{x \cdot y}{0.375 \cdot \|x^2\|^{0.65} \|y^2\|^{0.35}}$,

5. Przeprowadzenie ważonego głosowania - jeśli etykieta pojawiła się kilka razy (tzn. wśród kilku sąsiadów), to ich głosy były sumowane. Same wagi, raz jeszcze, zależały od odległości oraz pozycji wśród etykiet (jeśli sąsiad miał więcej niż jedną) i dobrane zostały heurystycznie,

6. Porównanie sugerowanych etykiet do rzeczywistej treści ogłoszeń, w celu wyboru 5 najlepszych etykiet. Analizowano tu każde słowo etykiety osobno. Waga ustalona w głosowaniu była odpowiednio zmniejszana, jeśli któreś słowo nie występowało w tytule ogłoszenia i nieznacznie zwiększana, jeśli słowo wystąpiło. Za najlepsze etykiety uznano te z najwyższymi wagami i zwrócono je jako ostateczną predykcję.

3.1 Inne zbiory danych

Nie użyto zbiorów danych innych niż udostępnione przez Organizatora.

4 Zależności

Lista wszystkich wykorzystanych bibliotek w `Python3(.6)`:

- `re`
- `numpy`
- `pandas`
- `scipy`
- `sklearn`
- `similaripy`[1] (v0.0.12, licencja MIT)

[1] `https://github.com/bogliosimone/similaripy`

5 Przykład użycia i opis kodu

Model został przygotowany w języku `Python3.6` w stylu podobnym do modeli w pakiecie `sklearn` (tzn. klasa z metodami `fit`, `predict`). Poniżej opisany został przykład użycia do predykcji etykiet dla 50.000 ogłoszeń ze zbiorem uczącym złożonym z 1.950.000 ogłoszeń. Kod zapewnia również funkcje do sprawdzenia wyniku (zgodnego z formułą oceniania Organizatora).

Warto podkreślić, że przytoczony w kodzie przykład użycia to przykład predykcji jedynie na części całego zbioru danych. Dla tej wielkości zbioru egzekucja kodu trwała mniej niż 30 minut z 8-rdzeniowym procesorem i 52GB pamięci RAM. Wynik (zgodny z formułą Organizatora) wyniósł tu 0.495226 i, jako że model jest całkowicie deterministyczny, powinien być taki sam na każdym komputerze.

```
import numpy as np
import pandas as pd
from SM_LabelPredictor import label_predictor

dataset = pd.read_csv("dataninja2019_ads_train.csv.gz",
                          compression="gzip", nrows=2000000)
dataset = np.array(dataset)[:, [1,2,-1]]
train_size = 1950000

train_labels= dataset[:train_size, -1]
test_labels = dataset[train_size:, -1]
train_set= dataset[:train_size, :-1]
test_set = dataset[train_size:, :-1]

lp = label_predictor(verbose=True, k_neighbors=46)
lp.fit(train_set, train_labels)
answers = lp.predict(test_set)

lp.rate_answers(answers, test_labels)
```

Kolejne kroki w kodzie zostały opisane poniżej:

1. Zaimportowano model za pomocą komendy `import label_predictor_SM`,
2. Utworzono zbiory treningowe i testowe (można wykorzystać model bez danych testowych, wtedy jednak bez funkcji oceny rozwiązania),
3. Utworzono instancję klasy podając argumenty `verbose` (`True` lub `False`). Z wartością `True` model zwracać będzie aktualny postęp. Argument `k_neigbors` warunkuje z ilu najbliższych sąsiadów model wybierze etykiety,

4. Dostosowano model do danych treningowych, używając metody `fit` z parametrami odpowiednio: 2-kolumnowe dane treningowe oraz wektor poprawnych etykiet,

5. Przewidziano etykiety dla danych testowych. Metoda `predict` zwraca sugerowane odpowiedzi. Dla pełnego zbioru danych testowych i treningowych ten krok potrwa dość długo (zależnie od liczby procesorów, około 5-10 godzin) i wymagać będzie dużo pamięci RAM (nawet 100GB),

6. Użyto metodę `rate_answers`, żeby otrzymać informację o skuteczności modelu (tylko jeśli posiadamy poprawne etykiety).

W przygotowanym modelu jako jedyny parameter dostępna jest liczba szukanych sąsiadów, w rzeczywistości jednak podobnych numerycznych parametrów jest o wiele więcej (około 20). Wszystkie ich wartości zostały heurystycznie zoptymalizowane do danych konkursowych. W celu dopasowania do danych z innego rozkładu, parametry te (i wiele innych nienumerycznych) musiałby zostać odpowiednio dopasowane.

6 Dodatkowe komentarze

Nowoczesne podejścia do przetwarzania języka naturalnego w tym przypadku nie sprawdziły się. Próbowano wykorzystać wektoryzację ogłoszeń za pomocą podejść Word2Vec oraz FastText, ale żadne z nich nie dorównało rzadkiej macierzy TF/IDF. Rozważano również użycie modyfikacji TF/IDF, tj. macierzy BM25 i BM25 plus, jednak w praktyce działały one nieco gorzej niż oryginalny pomysł. Być może modyfikacja BM25f (dla tekstów ustrukturyzowanych) zadziałałoby jeszcze lepiej, nie zostało to jednak sprawdzone.

Czasami gdy ze zbioru najbliższych sąsiadów unikalnych etykiet nie wystarczało (tzn. było ich mniej niż 5), to trzeba było ten zbiór uzupełnić. Próby uzupełnienia go wykorzystujące wyżej wymienione nowoczesne podejścia sprawdziły się gorzej niż proste pomysły. W tym przypadku najlepiej było użyć najpopularniejszych etykiet z całego zbioru treningowych etykiet albo ze zbioru najbliższych etykiet wyciągnąć pojedyncze słowa (np. `'spodnie_ciazowe'` → `'spodnie', 'ciazowe'`) i zasugerować je jako etykiety.

Próbowano poprawić błędy (literówki) w ogłoszeniach i tytułach do odpowiedników ze słownika (zbudowanego ze słów występujących w etykietach), co prawdopodobnie podniosłoby skuteczność modelu. Okazało się to jednak zbyt wymagające obliczeniowo.

Konkursowym etykietom wiele można zarzucić: czasami w zupełności nie pasowały do ogłoszenia, a czasami zawierały dziwne znaki (podobne do np. `'spodnie_ciazowe\\,15'`). Uwzględnienie tego w modelu okazało się być kluczowe! Innymi słowy: model musi być w stanie nadawać również te wadliwe etykiety.

Podsumowanie czynników, które były kluczowe do osiągnięcia ostatecznego wyniku:

- niepoprawianie etykiet ze zbioru treningowego, a poprawienie ich dynamicznie dopiero w ostatnim kroku - podczas wyszukiwania pojedynczych słów z etykiety w tytule ogłoszenia,
- ograniczenie słownika do słów ze zbioru treningowych etykiet (rozdzielonych, tzn. z usuniętymi znakami _). W przypadku zadania konkursowego, interesuje nas podobieństwo w kontekście etykiet, a nie to czy ogłoszenie rozpoczęło się od „Witam" albo „Dzień dobry",
- staranny i czasochłonny dobór parametrów modelu, szczególnie w ostatniej części - podczas wykluczania etykiet zawierających słowo niewystępujące w tytule ogłoszenia.

6.1 Zastrzeżenie

Obliczenia na pełnym zbiorze danych wykonywano w chmurze na wirtualnej maszynie z 24 procesorami i 300GB pamięci RAM i nie jest gwarantowane, że kod zadziała z mniejszą ilością pamięci. Dla przykładu - 96GB RAM nie wystarczyło na przetworzenie pełnego zbioru danych, tzn. 1.5 mln danych testowych i 5 mln danych treningowych.