Warsztaty z technik uczenia maszynowego

Rekomendacje produktowe na podstawie danych z poprzednich transakcji

Mikołaj Kida
Yelyzaveta Liubonko
Agnieszka Orzechowska
Wojciech Trybulec

---

1 Opis tematu

H&M Group to rodzina marek i firm z 53 rynkami internetowymi i około 4850 sklepami. Ich sklepy internetowe oferują kupującym szeroki wybór produktów. Przy zbyt wielu możliwościach klienci mogą mieć trudność ze znalezieniem tego, co ich interesuje lub czego szukają.Zagubiony klient może ostatecznie nie dokonać zakupu. Aby poprawić wrażenia z zakupów, kluczowe znaczenie mają rekomendacje produktów. Co ważniejsze, pomaganie klientom w dokonywaniu właściwych wyborów ma również pozytywny wpływ na zrównoważony rozwój, ponieważ np. zmniejsza ilość zwrotów.

Celem tego projektu jest opracowywanie narzędzia pozwalającego na znajdowanie rekomendacji produktowych dla danego klienta na podstawie danych z poprzednich transakcji, a także metadanych klientów i produktów. Do dyspozycji mamy proste dane jak rodzaj odzieży, czy wiek klienta, dane tekstowe z opisów produktów, a także dane obrazów ze zdjęciami odzieży.

2 Podział zadań

Nazwisko	Rola w zespole
Orzechowska Agnieszka	Obróbka danych
Liubonko Yelyzaveta	Wstępna analiza danych
Kida Mikołaj	Implementacja w chmurze
Trybulec Wojciech	Budowa modelu

3 Założenia techniczne

3.1 Język programowania

Aplikacja zostanie napisana w języku Python. Wstępnie zakładamy wykorzystanie biblioteki TensorFlow oraz Keras do stworzenia modelu/modeli oraz skorzystanie z usług chmurowych AWS w celu przeprowadzania obliczeń. Wybór platformy AWS jest motywowany znacznym rozmiarem danych wejściowych, którymi dysponujemy.

4 Dane

Są dostępne zdjęcia ubrań oraz trzy bazy danych: *articles*, *customers* i *transactions_train*.

W tabeli *articles* z 25 kolumnami mamy informacje o produktach takie jak:

- *article_id* - unikatowy identyfikator produktu,
- *product_code* - 6-cyfrowy kod produktu,
- *prod_name* - nazwa produktu,
- *product_group_name* - nazwa grupy produktu (łącznie 19 grup),
- *colour_group_code* - kod koloru,
- *colour_group_name* - nazwa koloru,
- *section_no* - numer sekcji w jakiej znajduje się produkt,
- *section_name* - nazwa sekcji,
- *detail_desc* - opis produktu.

Kolejną tabelą, z którego będziemy korzystać jest *customers*. Jest to baza danych o 7 kolumnach, która daje następujące informacje o klientach:

- *customer_id* - unikalny identyfikator klienta,
- *FN* - zmienna binarna, która mówi czy klient otrzymuje Fashion News,
- *Active* - zmienna binarna, która mówi czy klient jest otwarty na komunikacje,
- *club_member_status* - status w klubie, trzy unikalne wartości (Active, Precreate, Left club),
- *fashion_news_frequency* - częstotliwość wysyłania komunikatów do klienta, trzy unikalne wartości (NONE, Regularly, Monthly),
- *age* - wiek klienta,
- *postal_code* - kod pocztowy.

W bazie danych *transactions_train* mamy pięć kolumn, o następujących wartościach:

- *t_dat* - data transakcji w formie YYYY-MM-DD (string),
- *customer_id* - identyfikator klienta,
- *article_id* - identyfikator produktu,
- *price* - kwota zamówienia,
- *sales_channel_id* - , dwie unikalne wartości (1 - sklep stacjonarny, 2 - online).

4.1 Wybór modelu

Zdecydowaliśmy się na implementację modelu Collaborative Denoising Auto-Encoders (CDAE). Jest to model często wykorzystywany przy zagadnieniu wyboru N-najlepszych rekomendacji.

CDAE to sekwencyjna sieć neuronowa z jedną warstwą ukrytą. Warstwa wejściowa i wyjściowa składają się z k neuronów, gdzie k to ilość produktów na których dokoujemy predykcji. Jako wektor wejściowy podajemy listę produktów, które zakupił użytkownik, a na wyjściu otrzymujemy produkty które będzie chciał on zakupić.

Szkolenie sieci polega na podzieleniu informacji o dokonanych transakcjach na dwie grupy. Pierwsza z nich stanowi grupę uczącą, na podstawie której próbujemy przewidzieć zakup przedmiotów z drugiej grupy.

Dokładny opis sieci można znaleźć w artykule "Collaborative Denoising Auto-Encoders for Top-N Recommender Systems" [1].

5 Implementacja

5.1 Opis aplikacji

Aplikacja składa się z następujących części

- skrypt do preprocessingu *preprocess_data.pyskryptgwnymain.py*
- plik konfiguracyjny *configuration.json*
- implementacja modelu *models/CDAE.py*
- funkcje pomocniecze *functions/utility.py*

5.2 preprocess_data.py

Skrypt służy do przygotowania danych do uczenia. W pierwszym kroku wybieramy ustaloną w pliku konfiguracyjnym ilość artykułów (niestety nie udało nam się przetestować sieci dla wszystkich, czy nawet większości produktów, ze względu na ograniczenia sprzętowe). Następnie subsetujemy pozostałe zbiory danych, aby finalnie otrzymać transakcje zawierające tylko wybrane w kroku pierwszym produkty. Na koniec wyciągamy z danych o transakcjach potrzebne kolumny (client_id oraz article_id) i zamieniamy hashe client_id na liczny całkowite.

5.3 main.py

Skrypt główny, który tworzy instancję klasy CDAE i inicjuje proces uczenia z wybranymi w konfiguracji parametrami. O ile to możliwe przy uczeniu wykorzystywany jest procesor graficzny.

5.4 Konfiguracja

Konfiguracja znajduje się w pliku configuration.json. Na chwilę obecną można wybrać wartość hiperparametru *epoch*,nazwę pliku zawierającego spis transakcji (transaction_table_name), procent danych, który zostanie użyty do testowania (test_size), nazwy kolumn w zbiorze z transakcjami (column_names) oraz ilość produktów, które zostaną wybrane w preprocessingu (preprocessing_n).

5.5 funkcje pomocniecze *functions/utility.py*

Plik utility.py zawiera funkcje pomocnicze odpowiadające za wczytanie i zapis danych. Zawiera on funkcję tworzącą słownik do mapowania id produktów na kolejne liczby naturalne, co pozwala później w łatwiejszy sposób dopasować model. Aby możliwe było odtworzenie informacji, które produkty zarekomendował model, klasa utility.py zawiera funkcję pozwalającą na eksport słownika do pliku tekstowego.

5.6 Klasa CDAE

Główna klasa reprezentująca sieć CDAE. Klasa służy do tworzenia oraz trenowania sieci. Pozwala na konstrukcję, trening oraz zapisanie wytrenowanego modelu. Na wejściu klasa przyjmuje szereg hiperparametrów.

6 Wyniki

Przeszkoliliśmy model ograniczając się jedynie do 700 artykułów (większa ilość skutkowała problemami z dostępną pamięcią). Dla takiej ilości artykułów oraz 20 epoch otrzymaliśmy następujące parametry:

```
 ------------------------
precision@10:0.04147960866219633
recall@10:0.19218729007120788
precision@5:0.03066212304422703
recall@5:0.13494009013960645
map:0.11547530430899809
mrr:0.11721019865043224
ndcg:0.19848851521173325
ndcg@5:0.09749423374080997
ndcg@10:0.1170165871442649
------------------------
```

Jak można zauważyć wartość MAP (miara wykorzystywana jako kryterium oceny na Kaglu) dla naszego modelu wynosiła 11.5% co jest wynikiem ponad 3 razy lepszym niż najlepszy wynik wśród osób pracujących nad tym samym problemem. Należy jednak pamiętać, że oficjalne wyniki bazują na całych danych, a nie na wycinku.

7 Wnioski i możliwości rozwoju

Rodzaj rozwiązania, które wybraliśmy okazał się problematyczny ze względu na ograniczenia sprzętowe. Szkolenie sieci o zaprezentowanej architekturze zużywa duże zapasy pamięci. Przy wyborze 700 produktów zużycie RAM przekraczało 26GB. Nawet wykorzystanie usługi Google Cloud Pro nie dostarczyło wymaganej ilości zasobów.

Sieć pomimo wysokiej skuteczności w zagadnieniu rekomendacji posiada ograniczenie związane ze skalowaniem. Dla większych ilości produktów sieć ta staje się bezużyteczna.

W przypadku naszego problemu można by było spróbować zaimplementować inne modele, które lepiej skalują sie z liczbą produktów, na przykład architektura sieci neuronowych zaproponowanych w [2]. Ta architektura została użyta przez YouTube do implementacji ich systemu rekomendacji.

Warto również przetestować wpływ hiperparametrów na efekty szkolenia. W naszym podejściu mieliśmy ustalony *batch size* oraz *learning rate*, których zmiana mogłaby pozytywnie wpłynąć na osiągane wyniki.

Literatura

[1] https://alicezheng.org/papers/wsdm16-cdae.pdf

[2] https://dl.acm.org/doi/10.1145/2959100.2959190