Jan Jędrzejewski 21 stycznia 2023

Projekt IUM

Drugi etap

Modele rankingujące

W ramach rozwiązania postawionego problemu biznesowego stworzyłem trzy klasy modeli rankingujących.różniących się strategiami wybierania najlepszych piosenek do playlisty. Szczegółowy opis każdej metody znajduje się poniżej, a implementacja dostępna jest w folderze models kolejno w plikach o nazwach:

```
1) popularityModel.py
2) userProfileModel.py
3) targetModel.py
```

Schemat modeli - metody publiczne

1) Inicjalizacja obiektu modelu
   `model = Model()`
2) Podanie danych
   `model.fit(users, tracks, ...)`
3) Stworzenie playlisty
   `model.getPlaylist([101, 102])`
   a) wejście - lista id użytkowników
   b) wyjście - lista id piosenek w playliście
4) Stworzenie playlisty + ranking
   `model.getPlaylist_with_ranks([331, 332])`
   a) wejście - lista id użytkowników
   b) wyjście - lista id piosenek i ranking
5) Uszeregowanie zadanej listy piosenek
   `model.rank_tracks_for_users([401, 402], [“qwert, “asdfg])`
   a) wejście - lista id użytkowników
   - lista id piosenek
   b) wyjście - uszeregowana lista piosenek według metody rankingującej

System rankingowania

Model popularnościowy oraz model parametrów utworów dla każdej piosenki w zbiorze danych obliczają w zakresie [0, 100] wartość przypasowania utworu do playlisty - zero dla najgorszych utworów, 100 dla utworów najlepiej pasujących do użytkownika. Utwory są uszeregowane malejąco względem tej wagi i do docelowej playlisty zostaje wybranych n najlepszych piosenek. Model docelowy w swojej metodzie łączy dwie metody, zatem jego wynik jest w zakresie [0, 200] - maksymalnie po 100 za popularność i za dopasowanie do odtwarzanych piosenek.

Model popularności i ulubionych gatunków muzyki

Dane dla modelu:

- użytkownicy
- piosenki
- artyści

Model nie wykorzystuje informacji o logach z sesji, bazuje na ulubionych gatunkach muzycznych użytkowników, gatunkach przypisanych do artystów oraz na popularności piosenek.

Sposób rankingowania

Dla każdego z użytkowników pobierana jest lista ulubionych gatunków muzycznych. Jeżeli któryś gatunek się powtarza to dostaje wagę odpowiadającą liczbie wystąpień, jeśli gatunek występuje tylko raz dostaje wagę jeden. Dla każdego utworu dopisujemy gatunki muzyczne które odpowiadają artystom, którzy wydali utwory. Dla każdej piosenki wybierana jest popularność i modyfikowana na podstawie wystąpień wśród ulubionych gatunków muzycznych użytkowników.

Funkcja rankingująca - popularność

$popularność = popularność + [(100 - popularność) * poziomPopularności * współczynnik$

Gdzie:

- $poziomPopularności$ - liczba wystąpień ulubionych gatunków muzycznych użytkowników wśród gatunków przypisanych do piosenki
  - dla braku wystąpień przyjmuje wartość -0.1, aby ukarać piosenki które nie zawierają pożądanych gatunków
- $współczynnik$ - obliczany wzorem $\frac{1}{2 * liczba\ Użytkowników}$ - zależny od liczby użytkowników, jeżeli każdy u wszystkich użytkowników dwukrotnie wystąpi pożądany gatunek muzyki, utwór przyjmuje popularność na poziomie około 99 punktów
- Popularność nigdy nie przekroczy poziomu 100, oraz daje możliwość mniej popularnym piosenkom “nadgonić” jeżeli wiele gatunków użytkowników się pokrywa.

Model parametrów utworów

Dane dla modelu:

- użytkownicy
- piosenki
- sesje

Model nie wykorzystuje danych o artystach. Używa szczegółowych parametrów piosenek oraz historię przesłuchanych przez użytkownika utworów, wraz z wydarzeniami podczas odsłuchu.

Sposób rankingowania

Dla każdego utworu jest wyznaczany zmiennoprzecinkowy wektor z listy parametrów piosenek. Wykożystane parametry utworów to: "duration_ms", "release_date", "danceability", "energy", "key", "loudness", "speechiness", "acousticness", "instrumentalness", "liveness", "valence", "tempo". Dla podanych użytkowników, wyznaczany jest wspólny uśredniony wektor piosenek których wcześniej wysłuchali. Jest to średnia ważona dla każdego z parametrów osobno. Wagami są reakcje użytkownika na utwór: “like”:2, “play”:1, “skip”: -1.

Funkcja rankingująca - odległość

Ranking piosenek odbywa się za pomocą porównania odległości wektorów piosenek do uśrednionego wektora użytkowników za pomocą podobieństwa kosinusowego.

$$\text{cosine similarity} = S_C(A,B) := \cos(\theta) = \frac{\mathbf{A}\cdot\mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

Ta metoda zwraca wyniki z przedziału [-1, 1], w którym 1 jest dla wektorów identycznych, a -1 dla wektorów przeciwnych. Z charakterystyki danych w tym zadaniu modelowania, wektory nigdy nie będą przeciwne, dlatego faktyczne wyniki są zwracane z przedziału [0, 1]. Na potrzeby wyrównania poziomu odległości z poprzednią metodą rezultat prawdopodobieństwa kosinusowego jest mnożony razy 100.

Model docelowy

Dane dla modelu:

- użytkownicy
- piosenki
- artyści
- sesje

Model docelowy wykorzystuje wszystkie dostępne zbiory danych.

Sposób rankingowania

Model docelowy łączy ideę modelu popularnościowego oraz modelu parametrów utworów. Osobno oblicza wartości rankingu popularności, jak i odległość wektora użytkowników, od wektora utworu.

Funkcja rankingująca

Piosenki są rankingowane na podstawie sumy popularności i odległości danego utworu

```
ranking = track["popularity"] + track["distance"]
```

Testowanie działania modeli

Do zaprezentowania działania modeli w notebookach do testowania, dla przykładowych danych wejściowych uruchomiono podstawowe funkcje modeli i wyświetlono rezultaty. Poniższe nazwy plików bezpośrednio odpowiadają klasom modeli:

```
1) test-popularityModel.ipynb
2) test-userProfileModel.ipynb
3) test-targetModel.ipynb
```

Sprawność i porównanie modeli

Do porównywania modeli została użyta funkcja `rank_tracks_for_users()` dla każdego z modeli oraz metoda Top-N accuracy, w przypadku tego zadania Top-10, ponieważ standardowa długość playlisty to dziesięć piosenek. Metoda bazuje na wybraniu jednej piosenki którą użytkownik wysłucha, ale o której model nie wie (piosenka ze zbioru testowego logów z sesji), oraz stu losowych piosenek. Następnie model rankinguje podane piosenki i rejestrujemy pozycję tej jednej przykładowej piosenki, jeśli jest wśród 10 najwyżej rankingowych to zaliczamy te wystąpienia. Iloraz liczby wystąpień w Top-N dzielimy przez liczbę wszystkich przykładów aby otrzymać dokładność modelu.

W notebooku `evaluation.ipynb` dla każdego z modeli przeprowadzono przykładowe wyliczenie trafności modeli za pomocą tej metody. Na końcu przeprowadzono również porównanie wszystkich modeli dla tych samych danych i uśredniono wyniki.

Niestety powyżej opisana metoda porównywania modeli nie dała oczekiwanych rezultatów i dla każdego modelu zwracała wartości bliskie 10%, co odpowiadałoby rozkładowi jednostajnemu wśród losowo rankingowych piosenek.

Prawdopodobnie do rzeczywistego porównania modeli należałoby użyć innej metody porównywanie, lub przetestować model na prawdziwych użytkownikach.

Aplikacja w formie mikroserwisu

Opis działania

Aplikacja terminalowa pozwala użytkownikowi zalogować się do systemu poprzez podanie numeru id użytkownika. Dostępne są trzy podstawowe czynności:

- 1- Wygenerowanie nowej playlisty
- 2 - Wyświetlenie wcześniej wysłuchanych utworów przez użytkownika
- 3 - wylogowanie

Po wciśnięciu klawisza 1 jest możliwość dodawania użytkowników do generowaniej playlisty wpisując ich id, proces można zakończyć wciskając 0. Następnie dla każdej piosenki wyświetlany jest tytuł oraz artysta i użytkownik ma do wyboru trzy operacje: 1-”like”, 2-”play”, 3-”skip”. Po zakończeniu “słuchania” playlisty użytkownik wraca do głównego menu.

Eksperymenty oraz testy A/B

Po wykonanych operacjach przez użytkownika do pliku `log.json` zapisywane są logi z sesji użytkowników. Dodatkowo zapisywane jest pole “model_type” informujące z którego modelu korzystał użytkownik do wygenerowania playlisty. Reakcje użytkowników na piosenki w playliście można przeanalizować i określić skuteczność modelu.

Testy A/B polegają na tym że użytkownicy nie wiedzą że obecnie są na nich przeprowadzane testy, aby nie miało to wpływu na rezultaty. W aplikacji wszyscy użytkownicy o id większym od 300 korzystają z podstawowego modelu, którym został wybrany Model parametrów utworów, a pozostali użytkownicy, czyli o id pomiędzy 100 a 300 korzystają z Modelu docelowego.

Uruchomienie aplikacji

Aby uruchomić aplikację należy z poziomu folderu głównego projektu wykonać komendę:

```
python app.py
```

Wszystkie kody źródłowe aplikacji znajdują się właśnie w pliku app.py.

Podsumowanie

Stworzenie modelu rankingującego wymaga opracowania możliwie najlepszej metody rankingującej do porównywania ze sobą rekordów na podstawie dostępnych danych. Nie są łatwe do przetestowania i zmierzenia skuteczności, ponieważ nie jesteśmy w stanie w 100% przewidzieć co spodoba się odbiorcy,

W ramach tego projektu aby osiągnąć jeszcze lepszy model mógłbym zastosować metodę filtrowania zespołowego, która polega na polecaniu tych piosenek. którzy inni użytkownicy o podobnej historii sesji, chętnie słuchali.

Do zbadania jakości modeli rankingujących metoda Top-N nie przyniosła żadnych sensownych rezultatów, powinienem wykorzystać inne metody porównywania modeli rankingujących aby wynieść poparte wynikami wnioski.

Z mojej perspektywy modele nie zwracają kompletnie losowych wartości, bardzo często proponują utwory podobne do siebie i podobne do odtworzeń użytkownika, lub utwory charakteryzujące ten sam gatunek muzyki. Podczas wykorzystania modeli na prawdziwych odbiorcach mogłyby dawać zadowalające rezultaty.

Źródła:

1) wykłady z przedmiotu IUM
2) https://www.kaggle.com/code/gspmoreira/recommender-systems-in-python-101
3) https://www.datacamp.com/tutorial/recommender-systems-python
4) https://en.wikipedia.org/wiki/Cosine_similarity