POLITECHNIKA RZESZOWSKA
im. IGNACEGO ŁUKASIEWICZA

Rzeszów 29.01.2023

Programowanie w języku Python – Video Game Recommender

Koczynasz Jonasz 166658

Złotek Sebastian 166721

Inżynieria i Analiza Danych

L06 FS-DI

WYDZIAŁ MATEMATYKI I FIZYKI STOSOWANEJ POLITECHNIKI RZESZOWSKIEJ

POLITECHNIKA RZESZOWSKA
im. IGNACEGO ŁUKASIEWICZA

Spis treści

1. Nazwa projektu i krótki opis

Video Game Recommender jak angielska nazwa wskazuje pozwala na polecenie gier komputerowych użytkownikowi mającemu możliwość wyboru do jakiej gry chce on takie rekomendacje dostać tzn. wybierając grę program wyświetla nam 6 podobnych tytułów, użytkownik sam może wybrać taki tytuł z listy. Program ten opiera się na porównywaniu słów kluczowych, tagów, nazw itp. Wszystko wyświetlane jest na stronie internetowej.

2. Opis bibliotek

Selenium - biblioteka, która pozwala na automatyzację przeglądarek internetowych w celach testowych, a także na scraping i automatyzację zadań. Obsługuje wiele przeglądarek internetowych, w tym Chrome, Firefox i Safari, i może być używana z wieloma językami programowania, w tym z Pythonem. Biblioteka zapewnia sposób na interakcję z witrynami internetowymi i wykonywanie czynności, takich jak klikanie przycisków, wprowadzanie tekstu i wysyłanie formularzy.

Pandas - biblioteka służąca do analizy danych. Pozwala na łatwe i szybkie przetwarzanie i analizę dużych zbiorów danych, umożliwiając łatwe wczytywanie i zapisywanie danych z różnych źródeł, takich jak pliki CSV, arkusze kalkulacyjne, bazy danych i inne. Biblioteka zapewnia wiele funkcji umożliwiających wygodne manipulowanie danymi, takich jak agregacje, grupowanie, łączenie i scalanie danych, a także funkcje wizualizacji danych. Pandas jest często używany w dziedzinie uczenia maszynowego i analityki danych.

NumPy - to biblioteka służąca do obliczeń naukowych. Zawiera wiele funkcji i narzędzi do pracy z macierzami i tablicami numerycznymi, takich jak:

- tworzenie i manipulowanie macierzami i tablicami
- wyznaczanie statystyk i obliczenia matematyczne
- integracja z innymi bibliotekami do obliczeń naukowych, takimi jak SciPy i Matplotlib

NumPy jest często używany w dziedzinie uczenia maszynowego i analizy danych, a także do obliczeń naukowych i technicznych.

NLTK - biblioteka służąca do przetwarzania języka naturalnego. Zawiera wiele narzędzi i funkcji do analizy i manipulacji tekstem, takich jak:

- tokenizacja, czyli dzielenie tekstu na mniejsze części, takie jak wyrazy i zdania
- tagowanie części mowy, czyli określanie funkcji gramatycznej każdego słowa
- lematyzacja, czyli łączenie wszystkich form danego słowa z jego podstawową formą
- analiza emocjonalna, czyli określanie nastroju tekstu

NLTK jest często używany w dziedzinie uczenia maszynowego, analizy tekstu i badań naukowych.

Scikit-learn (sklearn) - biblioteka służąca do uczenia maszynowego. Zawiera wiele popularnych algorytmów uczenia maszynowego, takich jak:

- Regresja (np. linowa, wielomianowa)
- Drzewa decyzyjne i las drzew decyzyjnych
- K-najbliższych sąsiadów (KNN)
- SVM (Support Vector Machines)
- Naive Bayes

Biblioteka zawiera także funkcje do wyboru i oceny modeli, takie jak walidacja krzyżowa i grid search. Jest łatwa w użyciu i pozwala na szybką implementację i ocenę modeli uczenia maszynowego. Sklearn jest często używany w dziedzinie uczenia maszynowego i analityki danych.

Pickle - moduł, który pozwala na serializację i deserializację obiektów Python. Serializacja polega na zamianie obiektu w postaci binarnej, którą można zapisać na dysku lub przesłać przez sieć, a deserializacja polega na odtworzeniu tego obiektu z postaci binarnej. Pickle pozwala na zachowanie stanu obiektów, takich jak modele uczenia maszynowego, w sposób, w jakim były one zapisane, a następnie na ich ponowne użycie w przyszłości bez konieczności ponownego uczenia. Pickle jest łatwy w użyciu i pozwala na oszczędność czasu i zasobów w procesie uczenia maszynowego.

Streamlit - biblioteka, która umożliwia łatwe i szybkie tworzenie aplikacji webowych opartych na uczeniu maszynowym. Streamlit pozwala na łatwe i szybkie tworzenie interfejsów użytkownika (UI) i wizualizacji danych bez konieczności pisania kodu HTML, CSS i JavaScript.

Streamlit jest przeznaczony do użytku przez deweloperów i data scientistów, którzy chcą w łatwy i szybki sposób udostępnić swoje modele i wizualizacje danych w postaci aplikacji webowej. Biblioteka jest łatwa w użyciu i pozwala na dodawanie interaktywnych elementów, takich jak suwaki, pola tekstowe i przyciski, bez konieczności pisania kodu HTML, CSS i JavaScript.

Streamlit jest bardzo przydatny w tworzeniu prototypów aplikacji webowych i pozwala na szybką i łatwą prezentację wizualizacji danych i modeli uczenia maszynowego, co umożliwia uzyskanie szybkiej i łatwej opinii od ekspertów i użytkowników.

3. Opis używanych narzędzi

TfidfVectorizer - narzędzie w bibliotece sklearn służące do przekształcania tekstu w wektory cech. Jest to implementacja popularnej metody TF-IDF, która jest często stosowana w analizie tekstu i uczeniu maszynowym.

TF-IDF jest miarą, która określa ważność słowa w danym dokumencie w stosunku do całego zbioru dokumentów. Polega na obliczeniu częstotliwości wystąpienia słowa w danym dokumencie (TF) i odwrotnej częstotliwości wystąpienia tego słowa w całym zbiorze dokumentów (IDF).

TfidfVectorizer pozwala na łatwą konwersję tekstu na wektory cech, co jest niezbędne do użycia wielu algorytmów uczenia maszynowego. Można ustawić wiele opcjonalnych parametrów, takich jak stopwords (czyli słowa, które nie będą brane pod uwagę podczas konwersji), ngram_range (określenie zakresu n-gramów, czyli ciągów słów) i max_df (maksymalne dozwolone wystąpienie słowa w zbiorze dokumentów).

euclidean_distances - funkcja w scikit-learn, która oblicza odległość euklidesową pomiędzy punktami w danych wejściowych. Odległość euklidesowa jest jednym z najczęściej używanych sposobów mierzenia odległości między punktami w przestrzeni.

Funkcja euclidean_distances przyjmuje jako argumenty dwie macierze lub dwie tablice Numpy i zwraca macierz odległości euklidesowych pomiędzy wszystkimi punktami. Można ją wykorzystać w wielu różnych algorytmach uczenia maszynowego, takich jak k-najbliższych sąsiadów (k-NN), grupowanie (clustering) i analiza skupień (density estimation).

Warto zauważyć, że odległość euklidesowa nie jest odpowiednia do wszystkich problemów, ponieważ zakłada, że wszystkie cechy mają taką samą wartość i ważność. W niektórych przypadkach konieczne jest stosowanie innych miar odległości, takich jak odległość Mahalanobisa.

PorterStemmer - algorytm stemmingu, który jest używany do redukcji słów do ich rdzenia (stemu) w celu ujednolicenia i skompresowania tekstu. Algorytm ten jest opracowany przez Martina Portera i jest jednym z najbardziej popularnych i skutecznych algorytmów stemmingu w języku angielskim.

PorterStemmer wykorzystuje szereg reguł i transformacji, aby usunąć sufiksy i inne końcówki słów, takie jak przysłówki, rzeczowniki i czasowniki, pozostawiając tylko ich rdzeń. Dzięki temu tekst jest zredukowany do kluczowych słów, co może być przydatne w wielu zastosowaniach, takich jak analiza sentymentu, klasyfikacja tekstu i analiza tematów.

PorterStemmer jest dostępny jako część biblioteki Natural Language Toolkit (NLTK) w języku Python. Można go łatwo wykorzystać do stemmingu tekstu, wywołując funkcję PorterStemmer() i przekształcając słowa za pomocą metody stem().

WebDriver - interfejs API w bibliotece Selenium, który jest używany do automatyzacji testów i interakcji z przeglądarkami internetowymi. WebDriver umożliwia programistom wysyłanie poleceń i instrukcji do przeglądarki, takich jak klikanie przycisków, wprowadzanie tekstu, nawigowanie po stronach i wiele innych.

WebDriver jest kluczowym elementem Selenium i umożliwia automatyzację wszystkich głównych przeglądarek internetowych, w tym Google Chrome, Mozilla Firefox, Internet Explorer i Safari. Dzięki temu można łatwo testować różne strony internetowe i aplikacje internetowe bez ręcznego wykonywania tych samych czynności.

Aby korzystać z WebDriver w języku Python, należy najpierw zainstalować bibliotekę Selenium, a następnie utworzyć instancję WebDriver i wywołać na niej metody, takie jak get() do nawigacji do określonej strony internetowej i find_element_by_*() do wyszukiwania elementów na stronie. Można również użyć WebDriver do automatyzacji powtarzających się czynności, takich jak logowanie i wypełnianie formularzy.

4. Web-scraping

Importujemy potrzebne biblioteki oraz funkcje:

```
from selenium import webdriver
import time
from selenium.webdriver.support.select import Select
from selenium.webdriver.chrome.service import Service
```

```
with open('video_game.csv', 'w', encoding = 'utf-8') as file:
    file.write("game_title; description; game_genre; year_of_production \n")

driver_service = Service(executable_path='C:\webdrivers\chromedriver.exe')
driver = webdriver.Chrome(service=driver_service)

driver.get('https://www.imdb.com/search/title/?title_type=game&sort=user_rating,desc')

driver.maximize_window()

for page in range(651):
    game_title = driver.find_elements('xpath','//div[@class="lister-item-content"]/h3/a')
    description = driver.find_elements('xpath','//div[@class="lister-item-content"]/p[@class="text-muted"]')
    game_genre = driver.find_elements('xpath','//div[@class="lister-item-content"]/p/span[@class="genre"]')
    year_of_production = driver.find_elements('xpath','//div[@class="lister-item-content"]/h3/span[@class="lister-item-year text-muted unbold"]')
    director = driver.find_elements('xpath','//div[@class="lister-item-content"]/p[@class=""]')
    with open('video_game.csv', 'a', encoding = 'utf-8') as file:
        for x in range(len(game_genre)):
            file.write(game_title[x].text + ";" + description[x].text + ";" + game_genre[x].text + ";" + year_of_production[x].text + ";" + director[x].text + "\n")

        next_page = driver.find_element('xpath','//div[@class="desc"]/a[@class="lister-page-next next-page"]').click()
    file.close()

driver.close()
```

Kod pobiera informacje o grach video z IMDb i zapisuje je w pliku CSV. Używa biblioteki Selenium do automatyzacji przeglądania stron i uzyskiwania informacji za pomocą wyrażeń XPath.

Następnie tworzy plik "video_game.csv" i ustawia jego formatowanie na UTF-8. Następnie tworzy instancję serwisu webdrivera Chrome i ładuje stronę z wynikami wyszukiwania gier video na IMDb. Po maksymalizacji okna przeglądarki, kod przechodzi przez każdą ze stron wyników (651 stron).

Dla każdej strony, kod pobiera informacje o tytule gry, opisie, gatunku, roku produkcji i reżyserze za pomocą wyrażeń XPath. Informacje te są następnie zapisywane w pliku CSV. Na koniec, kod zamyka przeglądarkę i zamyka plik CSV.

5. Głowny kod programu

Importujemy potrzebne biblioteki:

```
import pandas as pd
import numpy as np
```

```
df = pd.read_csv('video_game.csv', encoding = 'utf8', error_bad_lines=False, sep=';', header=0, names=['game_title', 'description','game_genre','year', 'cast'] )
```

Odczytanie pliku z danymi

```
df['game_title2'] = df['game_title']
```

```
df['id'] = range(1, len(df) + 1)
df.set_index('id', inplace=True)
```

```
df[0:50]
```

```
df.dropna(inplace = True)
```

```
null_count = df['description'].isnull().sum()
```

```
df['year'] = df['year'].str.replace('\D', '', regex=True)
```

```
df.drop(df[df.description == 'Add a Plot'].index, inplace=True)
```

```
sub_to_remove = ['Directors:', 'Director:', 'Stars:', 'Star:']
df['cast'].replace(sub_to_remove, '', regex=True, inplace = True)
df['cast'] = df['cast'].apply(lambda x : x.replace('|',','))
```

Obróbka ramki danych, usunięcie niepotrzebnych znaków i pustych wartości.

```
df['cast'][1]
```

```
' Matthew Gallant, Bruce Straley ,  Ashley Johnson, Troy Baker, Hana Hayes, Jeffrey Pierce'
```

```
df['cast'] = df['cast'].apply(lambda x : x.replace(' ',''))
```

```
df['cast'][1]
```

```
'MatthewGallant,BruceStraley,AshleyJohnson,TroyBaker,HanaHayes,JeffreyPierce'
```

```
df['game_genre'] = df['game_genre'].apply(lambda x : x.replace(',',''))
df['game_genre'] = df['game_genre'].apply(lambda x : x.replace('.',''))
df['description'] = df['description'].apply(lambda x : x.replace(',',''))
df['description'] = df['description'].apply(lambda x : x.replace('.',''))


#### test
df['game_title'] = df['game_title'].apply(lambda x : x.replace(',',''))
df['game_title'] = df['game_title'].apply(lambda x : x.replace('.',''))
df['game_title'] = df['game_title'].apply(lambda x : x.replace(':',''))
df['game_title'] = df['game_title'].apply(lambda x : x.replace('-',''))
df['game_title'] = df['game_title'].apply(lambda x : x.replace('\'',''))
```

Dalsza część obróbki ramki danych i usuwania niepotrzebnych znaków, które mogą przeszkadzać w rekomendacji tytułów.

```
df['game_title'] = df['game_title'].apply(lambda x:x.split())
df['description'] = df['description'].apply(lambda x:x.split())
df['cast'] = df['cast'].apply(lambda x:x.split())
df['game_genre'] = df['game_genre'].apply(lambda x:x.split())
```

```
df
```

```
df['tags'] = df['game_title'] + df['description'] + df['cast'] # + df['game_genre']
```

```
new_df = df[['game_title2','tags']]
```

```
new_df = new_df.reset_index(drop = True)
```

```
new_df['id'] = range(1, len(new_df) + 1)
new_df.set_index('id', inplace=True)
```

```
new_df['tags'] = new_df['tags'].apply(lambda x: " ".join(x))
new_df['tags'] = new_df['tags'].apply(lambda x: x.lower())
```

Oddzielenie poszczególnych słów i utworzenie z nich nowej kolumny o nazwie tags, która będzie wykorzystana w rekomendacji, przekształcenie liter na małe oraz utworzenie indeksów do nowej ramki danych new_df.

```
import nltk
```

```
from nltk.stem.porter import PorterStemmer
ps = PorterStemmer()
```

```
def stem(text):
    y = []

    for x in text.split():
        y.append(ps.stem(x))

    return " ".join(y)
```

```
ps.stem('call')
```

```
call'
```

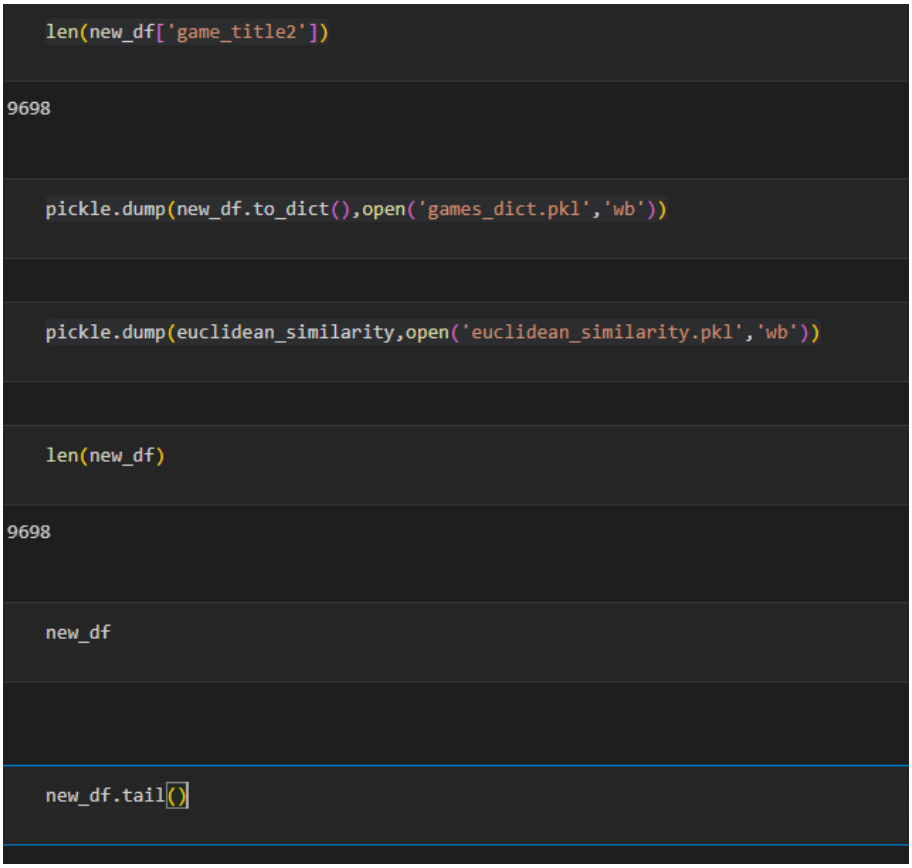
```
new_df['tags'].apply(stem)
```

Zastosowanie biblioteki nltk i funkcji PorterStreamer na kolumnie tags, co pozwala na „ucięcie końcówek” wyrazów dzięki czemu funkcja rekomendująca będzie mogła je porównywać.

```python
from sklearn.feature_extraction.text import TfidfVectorizer
tfidfvectorizer = TfidfVectorizer(stop_words='english',  ngram_range=(2, 2), min_df=6)


vectorizer  = tfidfvectorizer.fit_transform(new_df['tags']).toarray()


from sklearn.metrics.pairwise import euclidean_distances


euclidean_similarity = euclidean_distances(vectorizer)


def recommend(game):
    index = new_df[new_df['game_title2'] == game].index[0]
    distances = sorted(list(enumerate(euclidean_similarity[index])),reverse=False,key = lambda x: x[1])[1:6]

    for i in distances:
        print(new_df.iloc[i[0]].game_title2)
```

Kod importuje moduły TfidfVectorizer i euclidean_distances z biblioteki scikit-learn. Tworzy obiekt TfidfVectorizer z określonymi słowami zatrzymującymi w języku angielskim, bigramami i minimalną częstotliwością dokumentu wynoszącą 6. Następnie metoda fit_transform jest stosowana do kolumny 'tags' w dataframe 'new_df', aby wygenerować macierz TF-IDF, która jest konwertowana na tablicę numpy. Metoda euclidean_distances jest stosowana do wynikowej tablicy, aby wygenerować współczynniki podobieństwa euklidesowego.

Zdefiniowaniowaliśmy funkcję o nazwie 'recommend', która przyjmuje jako argument tytuł gry. Znajduje indeks gry w dataframe 'new_df', oblicza odległości od wejściowej gry do wszystkich innych gier w dataframe na podstawie współczynników podobieństwa euklidesowego, sortuje odległości w kolejności rosnącej i zwraca 5 najbliższych gier (bez uwzględnienia wejściowej gry) jako wynik.



```
len(new_df['game_title2'])
```

```
9698
```

```
pickle.dump(new_df.to_dict(),open('games_dict.pkl','wb'))
```

```
pickle.dump(euclidean_similarity,open('euclidean_similarity.pkl','wb'))
```

```
len(new_df)
```

```
9698
```

```
new_df
```

```
new_df.tail()
```

Kod oblicza długość kolumny 'game_title2' w dataframe 'new_df'. Następnie używamy biblioteki 'pickle' do serializacji i zapisania dataframe 'new_df' do pliku binarnego o nazwie 'games_dict.pkl'. Serializujemy i zapisujemy również tablicę numpy 'euclidean_similarity' do pliku binarnego o nazwie 'euclidean_similarity.pkl'. Na końcu kod zwraca ostatnie 5 wierszy dataframe 'new_df' przy użyciu metody 'tail'.

6. Tworzenie strony internetowej

Import potrzebnych bibliotek:

```
import streamlit as st
import pickle
import pandas as pd
```

```
games_dict = pickle.load(open('games_dict.pkl','rb'))
euclidean_similarity = pickle.load(open('euclidean_similarity.pkl','rb'))

games = pd.DataFrame(games_dict)
```

Wczytujemy zapisane wcześniej pliki binarne przy użyciu biblioteki 'pickle' i tworzymy dataframe 'games'

```
def recommend(game):
    index = games[games['game_title2'] == game].index[0]
    distances = sorted(list(enumerate(euclidean_similarity[index])), reverse=False, key=lambda x: x[1])[1:10]

    recommended_games = []
    for i in distances:
        recommended_games.append(games.iloc[i[0]].game_title2)
    return recommended_games

st.title('Game Video Recommender System')

selected_game_name = st.selectbox(
    'What game should I base my recommendations on?',
    games['game_title2'].values)

if st.button('Recommend'):
    recommendations = recommend(selected_game_name)
    for i in recommendations:
        st.write(i)
```

Kod definiuje funkcję 'recommend' przyjmującą jako argument tytuł gry, który ma być podstawą do wyboru rekomendowanych gier.

Funkcja używa indeksu gry podanej jako argument, aby znaleźć odległości euklidesowe do innych gier zapisanych w 'euclidean_similarity'. Następnie sortuje te odległości i wybiera dobraną ilość najbliższych gier.

Z wybranych gier tworzy listę 'recommended_games' i zwraca ją jako wynik funkcji.

Następnie tworzymy interfejs użytkownika za pomocą biblioteki 'streamlit', pozwalając użytkownikowi wybrać grę na podstawie której zostaną wybrane rekomendacje. Po wciśnięciu przycisku 'Recommend' funkcja 'recommend' jest wywoływana i wybrane rekomendacje są wyświetlane na ekranie.

7. Podsumowanie

Projekt "video game recommender" jest to system polecania gier wideo, który korzysta z biblioteki scikit-learn i algorytmu podobieństwa euklidesowego do wyznaczenia podobieństwa między różnymi grami. Tytuły gier są przekształcane na wektory cech za pomocą biblioteki 'TfidfVectorizer' i odległości między wektorami są obliczane za pomocą funkcji 'euclidean_distances'.

Interfejs użytkownika jest stworzony za pomocą biblioteki 'streamlit', pozwalając użytkownikowi wybrać grę, która ma być podstawą do wyboru rekomendowanych gier. Po wybraniu gry i wciśnięciu przycisku 'Recommend', system wywołuje funkcję 'recommend' i wyświetla listę x (dobrane przez programistę) rekomendowanych gier.

Wyniki są przechowywane w formacie binarnym, aby umożliwić szybkie ładowanie danych bez potrzeby ponownego obliczania odległości między grami.