Ilyefalvi Emese

Adatbányászat és gépi tanulás (beadandó)

Cél meghatározása

A Goodreads.com könyvekről szóló oldalon text review-kat lehet írni a könyvekhez és értékelni lehet őket ott egy 1-től 5-ig terjedő skálán. Az innen nyert több mint 11 000 rekordot tartalmazó tábla fellelhető a kaggle.com oldalon. Célom ezen adatbázis felhasználásával megvizsgálni, hogy az olvasmányok mennyi text review-t generáltak és mennyi az átlagos értékelésük. Egész pontosan szeretném megvizsgálni, hogy milyen kategóriák léteznek ezen a téren: Hogyan függ össze az, hogy egy könyv mennyi text review-t generál és az, hogy milyen értékelése. A kurzushoz megadott adatsorhoz képest a könyvek és értékeléseik közelebb állnak hozzám, ezért választottam más adatsort.

A feladathoz tartozó programkód és a megtisztított forrásfájlok megtalálhatók az alábbi github repozitóriumban: https://github.com/ilyefalvi/adatbanyaszat_es_gepi_tanulas_elte.git

Adatok megismerése és bemutatása

Az adatbázis egy vesszővel tagolt .csv-fájl, azaz egyetlen táblából áll, ez 11127 db rekordot tartalmaz. A fájl első sorában fellelhetők a mezőnevek is, amelyek a mezőkről szólnak:

- bookID: ez egy egyedi azonosító szám (egész szám)
- title: a könyv címe (szöveg)
- authors: A könyv szerző /-jellel elválasztva (szöveg/tömb)
- **average_rating**: A könyv értékeléseinek átlaga (0-től 5-ig terjed a skála) (tört szám)
- isbn: A könyv isbn száma (szöveg)
- isbn: A könyv 13 jegyű isbn-száma (szöveg)
- language_code: A könyv milyen nyelven írodott (rövid szöveg)
- num_pages: A könyv oldalszáma (egész szám)
- **ratings_count**: A könyvre hány értékelés érkezett (egész szám)
- **text_reviews_count**: A könyvre hány **szöveges** értékelés érkezett (egész szám)
- publication_date: A kiadás dátuma (dátum)
- publisher: a kiadás dátuma (szöveg)

Adatok előkészítése, előfeldolgozása

A beolvasás során a következő hibákat észleltem, ezeket javítottam:

- A csv-fájl vesszőkkel van tagolva, mégis néhány rekordban idézőjel nélkül szerepelnek vesszők, amelyek elrontják így a formátumok. Ezek a hibák és az általam alkalmazott átírások:
    - 3350. sor: "Sam Bass Warner, Jr." -> "Jr. Sam Bass Warner"
    - 4704. sor: "David E. Smith (Turgon of TheOneRing.net, one of the founding members of this Tolkien website)/Verlyn Flieger/Turgon (=David E. Smith)" -> "David E. Smith/Verlyn Flieger/Turgon"
    - 5879. sor: "James Wesley, Rawles" -> "Rawles James Wesley"
    - 8981. sor: "James, Son & Ferguson" -> "Brown/Son/Ferguson"
- A rekordok közt néhány dátum nem létező dátumra utalt:
    - 8182.sor: "11/31/2000" -> "11/30/2000"
    - 11100.sor: "6/31/1982" -> "6/30/1982"

Az adatokkal ez leszámítva nem volt egyéb probléma.

Alkalmazandó modell meghatározása

Az elemzés során unsupervised learninget, egész pontosan **K-means clustering**-et alkalmaztam a text_review és az avereage_rating mezők közötti összefüggés vizsgálatára. Ehhez külön függvényeket és programot írtam pythonban, az eredményt a LaTeX TikZ könyvtárának segítségével ábrázoltam. Utólag már persze látom, hogy scikit-tel gyorsabb lett volna, de így legalább gyakoroltam a programozást és jobben megértettem a modellt.

A továbbiakban a program kódja következik:

Book class

```python
from datetime import date

class Book:
    def __init__(b, t) -> None:
        b.bookID = int(t[0])
        b.title = t[1]
        b.authors = [cella.strip() for cella in t[2].split('/')]
        b.average_rating = float(t[3])
        b.isbn = t[4]
        b.isbn13 = t[5]
        b.language_code = t[6]
        b.num_pages = int(t[7])
        b.ratings_count = int(t[8])
        b.text_reviews_count = int(t[9])
        honap, nap, ev = t[10].split('/')
        b.publication_date = date(int(ev), int(honap), int(nap))
        b.publisher = t[11]

    def beolvas(fajlnev) -> list:
        lista = []
        with open(fajlnev, 'r', encoding="utf-8") as f:
            next(f)
            for sor in f:
                lista.append(Book([cella.strip() for cella in sor.split(',')]))
            return lista
```

Points class

```python
class Point:
    def __init__(p, nev, x, y) -> None:
        p.nev = nev
        p.x = float(x)
        p.y = float(y)
```

k_means_clustering.py függvényei

```python
from points import Point
from random import randint
from typing import Callable
from math import sqrt

def shuffled(t:list):
    l = t.copy()
    n = len(l)
    for i in range(n):
        r = randint(i, n-1)
        l[i],l[r] = (l[r], l[i])
    return l


def clusters_atlagos_abszolut_eltereseinek_osszege(clusters:dict, px:Callable, py:Callable):
    return sum(cluster_centroid_koruli_atlagos_abszolut_elterese(centroid, clusters[centroid], px, py) for centroid in clusters.keys())

def jobb_clusters(clusters1:dict, clusters2:dict, px:Callable, py:Callable):
    return clusters_atlagos_abszolut_eltereseinek_osszege(clusters1, px, py) > clusters_atlagos_abszolut_eltereseinek_osszege(clusters1, px, py)

def pont_rekord_tav_2D(p:Point, r, px:Callable, py:Callable) -> float:
    return sqrt((p.x-px(r))**2 + (p.y-py(r))**2)

def cluster_centroid_koruli_atlagos_abszolut_elterese(centroid:Point, cluster:list, px:Callable, py:Callable):
    return atlag([pont_rekord_tav_2D(centroid, rekord, px, py) for rekord in cluster])

def random_kivalaszt_visszateves_nelkul(table:list, K:int):
    return shuffled(table)[0:K]

def clustering_2d_with_random(K, table, nev, px, py):
    centroid_rekordok = random_kivalaszt_visszateves_nelkul(table, K)
    centroids = [Point(nev(crekord), px(crekord), py(crekord)) for crekord in centroid_rekordok]
    return clustering_2d(centroids, table, nev, px, py)
```

```python
def best_clustering_2d(N:int, K:int, table:list, nev: Callable, px:Callable, py:Callable):
    best_clusters, best_sum_of_distances = clustering_2d_with_random(K, table, nev, px, py)
    print(f'az {0}. próbálkozásban a távolságok összege {best_sum_of_distances} lett')
    for i in range(N-1):
        clusters, sum_of_distances = clustering_2d_with_random(K, table, nev, px, py)
        print(f'az {i+1}. próbálkozásban a távolságok összege {sum_of_distances} lett')
        if best_sum_of_distances > sum_of_distances:
            print(f'Ez jobb, mint a {best_sum_of_distances}, így cserélem')
            best_clusters = clusters
            best_sum_of_distances = sum_of_distances
    return best_clusters, sum_of_distances

def best_clustering_2d_for_elbow(N:int, K:int, table:list, nev: Callable, px:Callable, py:Callable) -> tuple[dict, list[float]]:
    clusters_lista = [None]*K
    sum_of_distances_lista = [None]*K
    for k in range(2,K):
        print(f'---- K = {k} ----')
        clusters, sum_of_distances = best_clustering_2d(N, k, table, nev, px, py)
        clusters_lista[k] = (clusters)
        sum_of_distances_lista[k] = (sum_of_distances)

    return clusters_lista, sum_of_distances_lista
```

```python
def clustering_2d(regi_centroidok:list[Point], table:list, nev: Callable, px:Callable, py:Callable):
    vege = False
    while not vege:
        # clusterek inicializálása a centroidok körül
        clusters = {}
        for centroid in regi_centroidok:
            clusters[centroid] = set()

        # pontok clusterekbe sorolása
        for rekord in table:
            legkozelebbi_centroid = get_legkozelebbi_centroid([(centroid, pont_rekord_tav_2D(centroid,rekord,px,py)) for centroid in clusters.keys()])
            clusters[legkozelebbi_centroid].add(rekord)

        #centroidok újraszámolása
        uj_centroidok = clusterek_sulypontjai(clusters, px, py)

        vege = ponthalmaz(uj_centroidok,3) == ponthalmaz(regi_centroidok,3)
        regi_centroidok = uj_centroidok
    return (clusters, sum([sum_of_distances(clusters, centroid, px, py) for centroid in clusters.keys()]))

def sum_of_distances(clusters, centroid:Point, px:Callable, py:Callable) -> float:
    return sum([pont_rekord_tav_2D(centroid, rekord, px, py) for rekord in clusters[centroid]])

def ponthalmaz(centroidok, pontossag:int):
    return {(round(centroid.x,pontossag), round(centroid.y,pontossag)) for centroid in centroidok}

def clusterek_sulypontjai(clusters:dict, px:Callable, py:Callable):
    return [Point(f'm{centroid.nev}', rekordok_atlaga(clusters[centroid], px), rekordok_atlaga(clusters[centroid], py)) for centroid in clusters.keys()]
```

```python
def rekordok_atlaga(cluster, p:Callable):
    return atlag([ p(rekord) for rekord in cluster])

def atlag(t):
    return sum(t)/len(t)

def get_legkozelebbi_centroid(centroid_tavok):
    best_centroid = centroid_tavok[0][0]
    best_tav = centroid_tavok[0][1]
    # print(f'eleinte: \n best_centroid: {best_centroid}\n best_tav: {best_tav}')
    for centroid, tavolsag in centroid_tavok:
        if tavolsag < best_tav:
            best_centroid = centroid
            best_tav = tavolsag
    return best_centroid
```

A TikZ-szel kapcsolatos kiíró parancsok

```python
def tikz_elbow(sum_of_distances_lista:list[float], fajlnev:str):
    nonementes = [elem for elem in sum_of_distances_lista if elem]
    print(nonementes)
    maksz = max(nonementes)
    K = len(nonementes)
    s = r'\begin{tikzpicture}[pont/.style={fill = black}]'
    s += f'\n\\pgfmathsetmacro{{\\K}}{{{K}}}\n'
    s += f'\n\\draw[step=1.0] (0,0) grid ({K},{str(maksz+1)});'
    for k in range(2,K):
        s+=f'\n\\node[pont](n{k}) at ({k}, {sum_of_distances_lista[k]}){{}};'
    s+= f'\n\\foreach \\k in {{1,2,...,{K}}}{{\n \\node[] at(\k, -.25){{\k}};}}'
    s+=r'\n\end{tikzpicture}'
    open(fajlnev, 'w', encoding='utf-8').write(s)


def centroid_to_tikzname(centroid:Point):
    return 'c' + centroid.nev.replace('m', '')

def rekord_to_tikz(rekord, centroid:Point, tikzlabel:Callable, tikznev:Callable, tikzszoveg:Callable, px:Callable, py:Callable):
    return f'\t\\node[rekord, {centroid_to_tikzname(centroid)} {tikzlabel(rekord)}]({tikznev(rekord)}) at ({px(rekord)},{py(rekord)}){{}};%{tikzszoveg(rekord)}\n'

def point_to_tikz(p:Point):
    return f'\t\\node[pont]({p.nev}) at ({p.x},{p.y}){{}};\n'
```

```python
def clusters_to_tikz(clusters:dict, tikzlabel:Callable, tikznev:Callable, tikzszoveg:Callable, px:Callable, py:Callable) -> str:
    clusterstilusok = ''

    szin = ['red','green','blue','cyan','magenta','yellow','black', 'white','gray','white','darkgray','lightgray','brown','lime','olive','orange','pink','purple','teal','violet', ]

    clusterstilusok = ',\n'.join([ f'{centroid_to_tikzname(centroid)}/.style={{fill={szin[i]}}}' for i, centroid in enumerate(clusters.keys())])

    s = '''\\begin{tikzpicture}[
pont/.style={fill = black},
rekord/.style={circle, draw=black},
''' + clusterstilusok +'\n]'

    for centroid in clusters.keys():
        for rekord in clusters[centroid]:
            s += rekord_to_tikz(rekord, centroid, tikzlabel, tikznev, tikzszoveg, px, py)
    for centroid in clusters.keys():
        s += point_to_tikz(centroid)
```

Vezérlő paraméterek meghatározása modelben

Először is leszűrtem az adatokat aszerint, hogy csak az 1000-nél több ratings_count-tal rendelkező adatokkal dolgozzak, így végül az adatok felével dolgoztam csak, de ez is elég számottevő az adatbázis mérete miatt.

Mivel a text_reviews_count 0 és 100000 közé esett, az average rating pedig 0 és 5 közé, ezért az elsőt 10000-zel osztottam, az utóbbit pedig 2-vel szoroztam, hogy egy 10x10-es négyzetben ábrázolhatóak legyenek az adatok.

Lefuttattam 20-szori samplingezéssel K=2,3,4,5,6,7,8,9 alkalommal a klaszterezést:

```python
legjobb_clusters, legjobb_sum_of_distances = best_clustering_2d_for_elbow(
    N = 20,
    K = 10,
    table = [book for book in books if book.ratings_count>1000],
    nev = lambda b: b.bookID,
    px = lambda b: b.average_rating*2,
    py = lambda b: b.text_reviews_count/10000,
    )
```

Aztán az „elbow method” során arra jutottam, hogy 5 klaszterre lenne érdemes bontani az adatokat. Itt tűnt úgy, hogy 5-nél több klaszter már nem csökkenti tovább olyan nagyon szignifikánsan a pontok centroidokhoz való közelségösszegeit. Itt aztán N=100-szor teszteltem és elmentettem a legkisebb távolságösszegű klaszterezést:

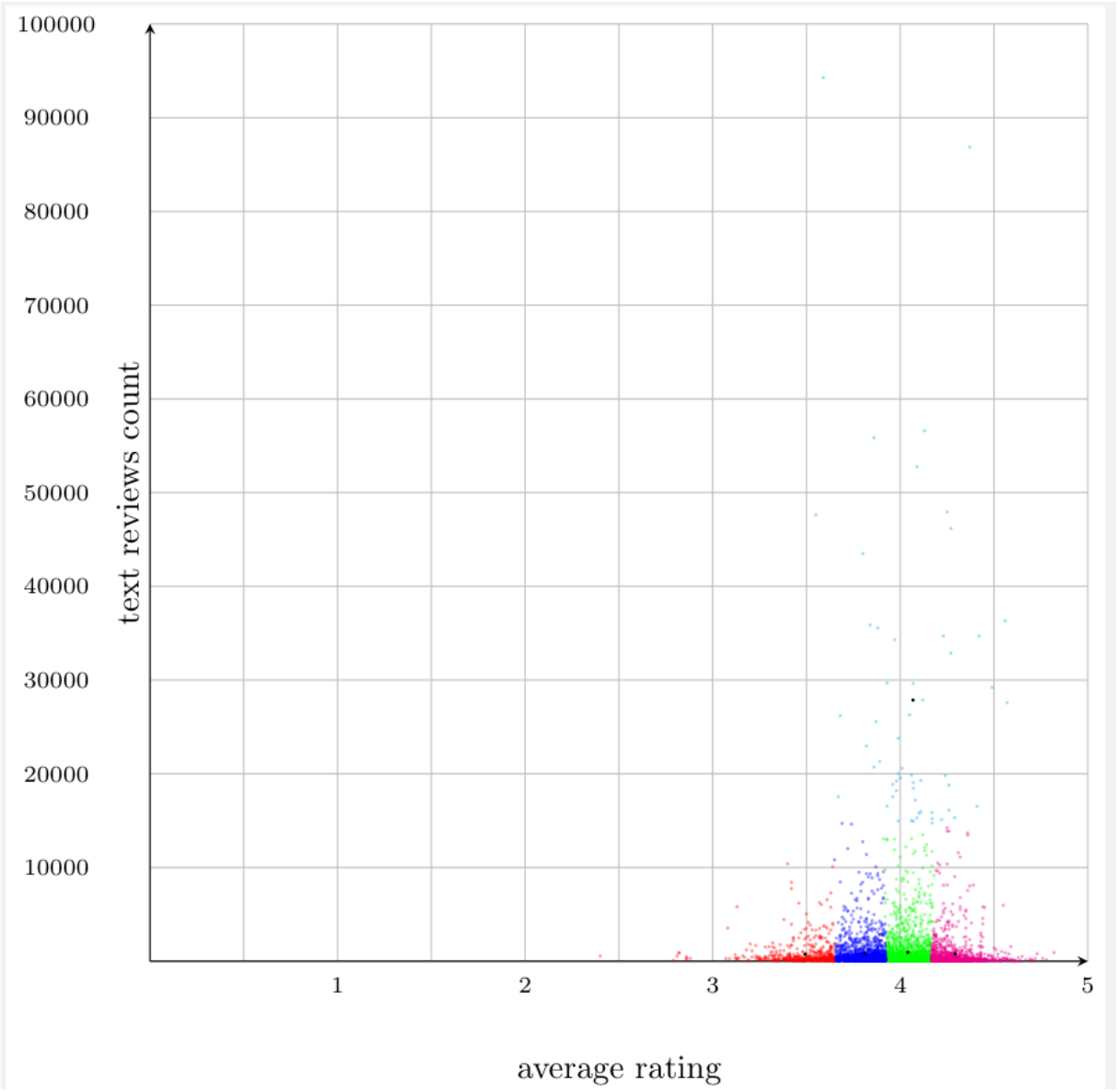
```python
legjobb_clusters, legjobb_sum_of_distances = best_clustering_2d(
    N = 100,
    K = 5,
    table = [book for book in books if book.ratings_count>1000],
    nev = lambda b: b.bookID,
    px = lambda b: b.average_rating*2,
    py = lambda b: b.text_reviews_count/10000,
    )
```

Kiírtam az adatokat

```
with open('output.tikz', 'w', encoding='utf-8') as f:
    f.write(clusters_to_tikz(
    clusters = legjobb_clusters,
    tikzlabel = lambda b: '',
    tiknev = lambda b: '',
    tikzszoveg= lambda b: b.bookID,
    px = lambda b: b.average_rating*2,
    py = lambda b: b.text_reviews_count/10000,
    ))
```

Eredmények bemutatása és részletes összehasonlítása

Az eredmény a következő grafikon lett:



A fekete pontok jelölik a centroidokat, az öt szín pedig az öt különböző clustert. Az ábráról az olvasható le, hogy azok a könyvek, amelyeket legalább ezren értékeltek, a 3-5 közötti tartományba esnek az átlagos értékelés terén. Mégis a 3,5-4,5 közötti könyvek azok, amelyek nagyobb számú text review-val bírnak (kék és zöld színnel jelölt klaszterek), illetve létezik egy ötödik cián színű klaszter is, amely a kifejezetten sok text review-t generáló könyveket tartalmazza. További érdekesség tehát, hogy a magas text-reviewt generáló könyvek általában nem tartoznak a legmagasabbra értékelt könyvek közé, noha nem is kapnak rossz értékeléseket.

Saját tapasztalatok levonása, saját gondolatok az elemzés kapcsán

Mindig is érdekelt, hogy milyen kapcsolat van a számszerű értékelés és a szöveges értékelés mennyisége között. Ez a felhőszerű cián klaszter most visszaigazolja azt, hogy érdemes volt ezt a k means clustering algoritmusával megvizsgálni, mert a nagy text review-t (tehát gondolatokat) generáló könyvek nem pusztán legjobb könyvek, hanem jól elkülönülő kategóriát alkotnak a könyvek között. Tehát azt lehetne mondani, hogy az oldalnak leginkább érdekes nagy text-reviewt generáló könyvek nem feltétlenül a „legjobb" könyvek egyben, hanem kell oda valami más is. A csúcstartó könyv egyébként a Twilight című könyv.

Személyes fejlődésemben nagyon sokat számított ennek a beadandónak az elkészítése. Egy barátom javasolta a tikz használatát a grafikonok készítésében, és most azt találtam, hogy nagyon szép képeket lehet vele készíteni – noha elég nehéz volt néha megtalálni benne a hibákat. Kicsit hamar választottam ugyanakkor irányt, mert utólag látom, hogy a scikit nevezetű libraryvel sokkal könnyebben meg lehetett volna oldani mindent. De így úgy érzem, hogy a saját függvények fejlesztésével jobban megértettem a modell matematikai hátterét is.