בינה מלאכותית – מערכות המלצה, תרגיל בית 3

אריאל סימון

חלק א - Non-personalized:

1. מימוש נאיבי של מערכת המלצה אישית הינו מיון לפי כל הדירוגים שהספרים קיבלו ע"י המדרגים, והחזרת k הספרים עם ההמלצות הכי טובות – על בסיס הדירוגים הכי גבוהים. מובן מיד ששיטה זאת אינה טובה מספיק – לפי מימוש זה, ספר שדורג בציון 5 מתוך 5 ע"י שני מדרגים בסך הכל, ייחשב טוב יותר מספר שדורג ע"י 100 אנשים וקיבל את הציון 4 מתוך 5. כלומר, עלינו להתחשב בנתון נוסף, סף של מספר הצבעות מינימלי כדי להיכנס לרשימה, ולכן נשתמש ב weighted average ratings.

במימוש שלנו למערכת המלצה מסוג Non-personalized, הוספנו עמודה בשם "num_of_users_that_vote_per_book". עמודה זאת מחשבת את מספר הקולות (מספר המצביעים) לכל ספר.

בנוסף, הוספנו עמודה בשם "average_rating". עמודה זאת מציגה את הדירוג הממוצע שהספר קיבל, כאשר חישוב זה נעשה ע"י פרמטרים נוספים שחושבו: סכום ההצבעות פר ספר, לחלק לסך המצביעים פר ספר.

כמו כן, חישבנו את הערכים m,c, כאשר:

m = מספר ההצבעות המינימלי הנדרש לספר.

c = ממוצע הדירוג עבור כל הספרים.

2. כתבנו פונקציה בשם get_simply_recommendation(k), המקבלת כקלט ערך k מהמשתמש, המייצג כמה פריטי המלצה נדרש להחזיר. הפונקציה מחזירה כפלט את k הספרים המומלצים, תוך ציון ה-id של הספר והציונים שקיבל.

להלן החזר הפונקציה עבור k=10 הספרים המומלצים:

```
>>> get_simply_recommendation(10)
     book_id                                                      book_name  counts  average_rating  weighted_rating
24        25                 Harry Potter and the Deathly Hallows (Harry Potter, #7)    1444        4.421745         4.338028
3          4                                                To Kill a Mockingbird    1747        4.364625         4.299843
101      102                                            Where the Wild Things Are     612        4.449346         4.273212
84        85                                                     The Giving Tree     815        4.364417         4.240309
49        50                                                Where the Sidewalk Ends     976        4.343238         4.239724
30        31                                                              The Help    1265        4.318577         4.238851
143      144  Unbroken: A World War II Story of Survival, Resilience, and Redemption     557        4.396768         4.221864
26        27               Harry Potter and the Half-Blood Prince (Harry Potter, #6)    1394        4.282640         4.213906
0          1                                 The Hunger Games (The Hunger Games, #1)    1764        4.238662         4.187383
132      133                          Anne of Green Gables (Anne of Green Gables, #1)     533        4.348968         4.181489
```

3. עבור מערכת המלצה Non-personalized שמתבססת על מקום מגוריו של המשתמש (place), סיננו את המידע לקבוץ חדש, כך שהוא מכיל רק users מאותו מקום מגורים שביקש המשתמש.

לאחר מכן, ביצענו weighted average ratings על המידע שסוכן.

כתבנו פונקציה בשם get_simply_place_recommendation(place, k), המקבלת 2 קלטים: מקום מגורים וערך k מהמשתמש, ומחזירה כפלט את k הספרים המומלצים, על בסיס מדרגים מאותו מקום מגורים.

להלן k=10 הסרטים המומלצים שהתקבלו עבור משתמש המתגורר בOhio:

```
>>> get_simply_place_recommendation('Ohio',10)
     book_id                                                  book_name  counts  average_rating     score
125      126                                     Dune (Dune Chronicles #1)      12        4.750000  4.367963
142      143                                  All the Light We Cannot See      15        4.600000  4.317786
143      144  Unbroken: A World War II Story of Survival, Resilience, and Redemption       8        4.750000  4.266087
23        24                 Harry Potter and the Goblet of Fire (Harry Potter, #4)      25        4.400000  4.249728
101      102                                     Where the Wild Things Are      15        4.466667  4.226877
461      490          Maus I: A Survivor's Tale: My Father Bleeds History (Maus, #1)       7        4.714286  4.213664
1219    1462                                       The Orphan Master's Son       7        4.714286  4.213664
876      983                                      Between the World and Me       7        4.714286  4.213664
118      119                                           The Handmaid's Tale      13        4.461538  4.199565
88        89                                           The Princess Bride      14        4.428571  4.190062
```

4. באופן דומה לסעיף 3, ביצענו סינון על הנתונים כך שנקבל רק נתונים רלוונטים, אך הפעם הסינון היה על users שדירגו ספרים מאותו גיל של המשתמש. לאחר מכן, ביצענו weighted average ratings על המידע שסונן.

כתבנו פונקציה בשם get_simply_age_recommendation(age, k), המקבלת 2 קלטים: גיל המשתמש וערך k מהמשתמש. הפונקציה מחזירה כפלט את k הספרים המומלצים, בהתאם לגיל המבוקש.

להלן k=10 הספרים המומלצים עבור משתמש בן 28:

```
>>> get_simply_age_recommendation(28,10)
     book_id                                                 book_name  counts  average_rating     score
24        25       Harry Potter and the Deathly Hallows (Harry Potter, #7)     186        4.413978  4.326251
3          4                                        To Kill a Mockingbird     216        4.365741  4.294203
84        85                                             The Giving Tree      99        4.444444  4.289614
88        89                                           The Princess Bride      69        4.449275  4.244702
132      133                Anne of Green Gables (Anne of Green Gables, #1)      73        4.410959  4.224914
49        50                                       Where the Sidewalk Ends     137        4.313869  4.216411
101      102                                     Where the Wild Things Are      78        4.371795  4.204680
69        70                                Ender's Game (Ender's Saga, #1)      93        4.344086  4.204095
30        31                                                      The Help     143        4.293706  4.202891
20        21  Harry Potter and the Order of the Phoenix (Harry Potter, #5)     169        4.272189  4.196385
```

חלק ב- Collaborative filtering user based:

5. מימשנו את האלגוריתם Collaborative filtering ע"פ user-base. את מטריצת החיזוי בנינו באמצעות הפונקציה build_CF_prediction_matrix(sim), כאשר sim הינו הקלט המתקבל מהמשתמש עבור מדד הדמיון – cosine, euclidean או jaccard.

6. כתבנו את הפונקציה get_CF_recommendation(user_id, k), כאשר k הינו ברגיל מספר הספרים שמעוניינים לקבל בחזרה כהמלצות, ו user_id הוא מזהה המשתמש. הפונקציה משתמשת בפונקציה build_CF_prediction_matrix(sim) ליצירת מטריצת פרדיקציה.

בהתאם להוראות של המתרגלת אסנת, השארנו בקוד את sim='cosine', כך שאם תתבצע קריאה ע"י צוות הקורס לפונקציה get_CF_recommendation(user_id, k), החישוב יבוצע ע"פ מדד הדמיון cosine.

7. מטריקות הדמיון מומשו כחלק מהפונקציות המובנות של Euclidean ו-Jaccard הקיימות בחבילה sklearn.metrics (באישור המתרגלת אסנת).

חלק ג' – contact based filtering

8. בחלק זה מימשנו את אלגוריתם ההמלצות contact based filtering. האלגוריתם משתמש בידע על הפריטים עצמם ע"י פיצ'רים (מאפיינים) מוגדרים, ויוצר וקטור דמיון לכל פריט עם שאר הפריטים האחרים. הפונקציה לבניית מטריצת הדמיון הינה build_cobtact_sim_metrix(), לפי פונקציית הדמיון cosine כנדרש בתרגיל.
הפיצ'רים בהם בחרנו להשתמש הינם Authors, Original publication year, Original title ו-Language code.

```
##### part 3 - contact based filtering #####

metadata = pd.read_csv('books.csv', low_memory=False, encoding="ISO-8859-1")
# remove bad rows - as Osnat did in her code.
metadata = metadata.drop([1613])
# choose features name:
features_metadata = {"book_id": metadata["book_id"], "authors": metadata["authors"],
                     "original_publication_year": metadata["original_publication_year"],
                     "original_title": metadata["original_title"], "language_code": metadata["language_code"]}
df_features_metadata = pd.DataFrame(features_metadata)
```

9. הפונקציה לקבלת ההמלצות שכתבנו הינה get_contact_recommendation(book_id, k).
K הפריטים שקיבלו את הציון הגבוה ביותר בוקטור הדמיון עבור ה-book_id המבוקש יחזרו אל המשתמש.

10. עבור הספר Twilight, בהינתן K=10, רשימת הספרים שמחזיר המודל הינה:

```
>>> get_contact_recommendation('Twilight', 10)
4051       The Twilight Saga: The Official Guide
51                                       Eclipse
72                                      The Host
2014    The Twilight Collection (Twilight, #1-3)
3058                 Twilight: The Graphic Novel
990                             The Twilight Saga
55                                  Breaking Dawn
48                       New Moon (Twilight, #2)
1386                                Eleven on Top
3142                                    Elsewhere
```

חלק ד' – מדדי הערכה

11. כתבנו את הפונקציות precision_k(k), ARHR(k), RMSE(), הרצנו כל אחת עם 3 מדדי הדמיון שלנו: cosine, Euclidean, jaccard.

להלן התוצאות שקיבלנו:

```
Python 3.8.7 (tags/v3.8.7:6503f05, Dec 21 2020, 17:59:51) [MSC v.1928 64 bit (AMD64)] on win32
>>> runfile('C:/Users/ariel.DESKTOP-6TR9210/PycharmProjects/ex3_all_scripts/Ariel_and_Linoy.py', wdir='C:/Users/ariel
 .DESKTOP-6TR9210/PycharmProjects/ex3_all_scripts')
>>> RMSE()
[0.9009798683709476, 0.9191466001886913, 0.9049450569632987]
>>> ARHR(10)
(0.3234761904761905, 0.03333333333333333, 0.32176190476190475)
>>> precision_k(10)
(0.08, 0.008, 0.08)
```

סדר ההדפסה משמאל לימין הינו 1.cosine, 2.euclidean, 3.jaccard.

בטבלה:

	Precision_k	ARHR	RMSE
Cosine	0.08	0.3234761904761905	0.9009798683709476
Euclidean	0.008	0.03333333333333333	0.9191466001886913
Jaccard	0.08	0.32176190476190475	0.9049450569632987

12. הסבר התוצאות:

מדדי ההערכה

על-פי התוצאות שהתקבלו, אנו רואים כי מדד ההערכה RMSE נותן את ההערכה הגבוהה ביותר למערכת ההמלצות. לעומת זאת, מדדי הARHR וה-Precision_k נותנים מדדים פחות גבוהים, כאשר ניתן לומר בכלליות כי הערכה ע"י precision_k הינה הנמוכה ביותר מבין ה-3.
הדבר הגיוני כי RMSE לא מתחשב ב-k שבדקנו (במקרה זה, k=10), אלא בודק את כל האפשרויות ולכן בסופו של דבר נקבל בו את הציון הגבוה יותר.
בשני מדדי ההערכה האחרים, ביצענו "סינון מקדים" של המשתמשים שאנו בודקים והדבר תורם לתוצאות הנמוכות. מבחינת ההבדל בין precision_k לבין ה-ARHR, הדבר נובע מאופן החישוב של שני המדדים:
בחישוב של precision_k, אנחנו בודקים את כמות הhits, מחלקים בk וסוכמים את כל התוצאות. למשל עבור k=10 ומשתמש שקיבל 3 hits, נחשב 3/10.
בחישוב של ARHR, אנחנו בודקים את כמות הhits ומחלקים לפי מיקום ה-hits ברשימת ההמלצות שנתנו. למשל, עבור k=10 ומשתמש שקיבל 3 hits, שברשימת ההמלצות שלנו עבורו היו במיקומים 2,5,8 – אנו נסכום 1/2 + 1/5 + 1/8.
ניתן לראות בדוגמא הנ"ל שהסכום שנקבל עבור משתמש עם 3 hits יהיה גבוה יותר בחישוב הARHR (0.3 למול 0.825)
הדבר יתקיים גם אם 3 הhits יהיו במיקומים "גרועים" כמו 8,9,10: בחישוב של ARHR נקבל 1/8+1/9+1/10 = 0.336, מה שעדיין גבוה יותר מהחישוב של precision_k ששווה 0.3.
כאשר סוכמים תוצאות כאלה עבור מספר משתמשים, ההבדל משמעותי אף יותר.

מדדי הדמיון

ניתן לראות כי בכל מדדי ההערכה שפירטנו קודם, מדדי הדמיון Cosine ו-Jaccard נותנים תוצאות דומות מאוד אחד לשני (תחת אותו מדד הערכה), ואילו Euclidean נותן תוצאה שונה.

במדד הRMSE – הEuclidean נתן תוצאה גבוהה יותר (במעט) משני המדדים האחרים, ואילו ב-ARHR וב-Precision_k הוא נתן תוצאה נמוכה פי 10 מהתוצאות של שני המדדים האחרים.

הדבר נובע, מחד, מההבדלים בין שלושת מדדי הדמיון, ומנגד - בהבדלים עצמם בין מדדי ההערכה (RMSE הכי גבוה, precision_k הכי נמוך)

הסיבה שהEuclidean נותן תוצאה שונה ביחס לשני מדדי הדמיון האחרים, היא שEuclidean מחשב דמיון <u>לפי המרחק בין וקטורי הדמיון במרחב</u>, חישוב שייתן תוצאה "רחוקה יותר" ביחס לcosine שמחשב מרחק פשוט בין נקודות, או לJaccard שמחשב דמיון ע"י מספר הפריטים המשותפים ששני משתמשים דירגו.