### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### **Query Kings-תרגיל 3**

**קישור לגיטהאב**:[**https://github.com/guybanbo/IR-Query-Kings.git**](https://github.com/guybanbo/IR-Query-Kings.git)

### שאלה 1:

דוגמה:

נניח שאנחנו מסווגים אנשים לפי הגובה שלהם לשתי קבוצות:

1. **קבוצה 1 (C₁):** אנשים נמוכים, עם גבהים 150 ס"מ ו-160 ס"מ.
2. **קבוצה 2 (C₂):** אנשים גבוהים, עם גבהים 180 ס"מ ו-190 ס"מ.

#### **חישוב מרכזי הקבוצות:**

* **מרכז C₁:** (150 + 160) / 2 = 155.
* **מרכז C₂:** (180 + 190) / 2 = 185.

### **דוגמת למידה חדשה:**

אדם שגובהו **170 ס"מ**.

#### **שלב הסיווג:**

* המרחק של 170 מ-C₁: |170 - 155| = 15.
* המרחק של 170 מ-C₂: |170 - 185| = 15.

המרחקים שווים, ולכן המסווג עלול לבחור באופן שרירותי (או לפי כלל כלשהו) לאיזו קבוצה לשייך אותו.

מדוע זו שגיאה?

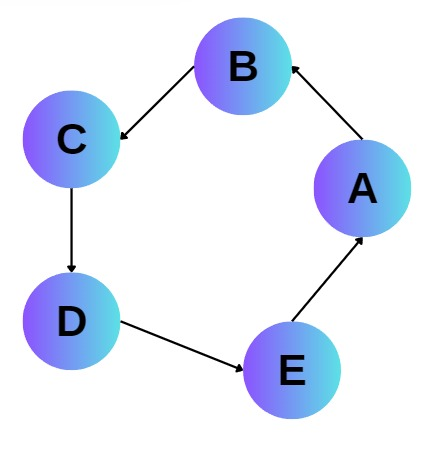
170 ס"מ זה גובה ממוצע שלא מתאים בבירור לאף אחת מהקבוצות (נמוך או גבוה). אבל מסווג Rocchio **חייב לבחור קבוצה**, ולכן עלול לסווג אותו כ"נמוך" (C₁) או "גבוה" (C₂), אף על פי שהאדם הוא בגובה בינוני.

הוכחת הטענה:

אם נסתכל במציאות, ברור ש-170 ס"מ לא נמוך וגם לא גבוה (כי הוא נמוך מ-180 ס"מ).  
מכאן שמסווג Rocchio נותן תשובה שגויה במצב שבו דוגמת הלמידה נמצאת באמצע בין הקבוצות.

שאלה 2:

א.



בגרף שבחרנו, דרגת הכניסה והיציאה של כל צומת היא 1. מכיוון שמכל צומת יוצאת קשת אחת בלבד לצומת אחרת, אם נחשב באופן איטרטיבי את ציוני ה-Authority וה-Hub של הצמתים לפי הנוסחאות שנלמדו, נראה שהערכים נשארים קבועים על 1.

לפי נוסחת העדכון של Authority, נסתכל על כל הצמתים שמחוברים לצומת הנוכחית באמצעות קשת נכנסת ונחבר את ערכי ה-Hub שלהם. כיוון שלכל צומת יש בדיוק קשת נכנסת אחת, סכום ערכי ה-Hub יתבסס על צומת אחת בלבד, שהערך שלה מתחילת החישוב הוא 1. לכן, ערך ה-Authority יישאר 1.

באופן דומה, בערכי ה-Hub נסתמך על סכום ערכי ה-Authority של הצמתים המחוברים לצומת באמצעות קשת יוצאת. גם כאן, לכל צומת יש קשת יוצאת אחת בלבד, ולכן ערך ה-Hub יתבסס על צומת אחת בלבד, שהערך שלה מתחילת החישוב הוא 1. לכן, גם ערך ה-Hub יישאר 1.

כך, בגרף שבו דרגת הכניסה והיציאה של כל צומת היא 1, ערכי ה-Authority וה-Hub נשארים קבועים ושווים ל-1 לאורך כל החישובים.

ב.

**Initial scores (all set to 1):**

Hub scores: h(A) = h(B) = h(C) = 1

Authority scores: a(A) = a(B) = a(C) = 1

**First iteration:** Authority update: ו

a(A) = h(A) + h(C) = 2

a(B) = h(A) + h(C) = 2

a(C) = h(A) + h(B) = 2

נרמול על ידי שורש של סכום ריבועי ערכי הAuthority(נרמול רגיל)

Normalize:

a(A) = =a(B) = a(C)

Hub update:

h(A) = a(A) + a(B) + a(C) = 3\*0.58 = 1.74

h(B) = a(C) = 0.58

h(C) = a(A) + a(B) = 2\*0.58 = 1.16

h(A) = , h(B) = , h(C) =

**Second iteration:**

Authority update:

a(A) = h(A) + h(C) = 0.8 + 0.54 = 1.34

a(B) = h(A) + h(C) = 1.34

a(C) = h(A) + h(B) = 0.8 + 0.27 = 1.07

Normalize: divide by

a(A) = 1.34/2.17=0.62 , a(B) = 1.34/2.17=0.62 , a(C) = 1.07/2.17=0.49

Hub update:

h(A) = a(A) + a(B) + a(C) = 2\*0.62 + 0.49 = 1.73

h(B) = a(C) = 0.49

h(C) = a(A) + a(B) =2\*0.62 = 1.24

Normalize: Sum ==2.18  
 h(A) = =0.79 , h(B) = =0.22 , h(C) ==0.57

**Third iteration:**

Authority update:

a(A) = h(A) + h(C) = 0.79 + 0.57 = 1.36

a(B) = h(A) + h(C) = 1.36

a(C) = h(A) + h(B) = 0.79 + 0.22 = 1.01

Normalize: =2.17

a(A) ==0.63 , a(B) = =0.63 , a(C) ==0.47

Hub update:

h(A) = a(A) + a(B) + a(C) = 2\*0.63 + 0.47 = 1.73

h(B) = a(C) = 0.47

h(C) = a(A) + a(B) = 2\*0.63 =1.26

Normalize: =2.19

h(A) = =0.79 , h(B) = =0.21 , h(C) ==0.58

הנרמול מחושב על ידי שורש של סכום הריבועים של ציוני authority או hub וכל ציון כזה מחולק בסכום הזה.

ניתן לראות שבאיטרציות 2 ו-3 הערכים מאוד קרובים בין האיטרציות(הבדל של 0.01-0.02) ומתכנסים כבר סביב ערכים מסוימים ולכן אין צורך בעוד איטרציות.

לכן, ציוני ה-hub וה-authority של כל דף הם:

a(A) =0.63 , a(B) =0.63 , a(C) =0.47

h(A) = 0.79 , h(B) = 0.21 , h(C) = 0.58

**שאלה 3**

1.את הזחלן והשאילתות מימשנו בפייתון והרצנו בGoogle Colab.

בפרויקט זה השתמשנו במספר טכנולוגיות מעניינות שתרמו לפיתוח ולמימוש הפתרון:

### **א) Firebase**

**השתמשנו במסד נתונים Firebase כדי לשמור את הדפים שהזחלן אסף.**

**לצורך ביצוע השאילתות, שלפנו משם את הדפים הרלוונטיים לכל שאילתא.**

### **ב)BeautifulSoup**

שימוש ב-BeautifulSoup אפשר לנו לחלץ את המידע שאנחנו רוצים מדפי HTML.

הזחלן השתמש בזה כדי לחלץ קישורים ואת הכותרות של דפים.

בכל שאילתא השתמשנו בספרייה זו כדי לחלץ את הקיצורים ואת המידע הרלוונטי לה.

ג) **openpyxl**- ספריית פייתון ליצירת קבצי אקסל. השתמשנו בספריה זו כדי לייצא לאקסל את הדפים שהזחלן אסף ואת כל המידע הרלוונטי והחישובים שנאספו ונעשו בשאילתות.

2. שאילתות 1-2 רצו 2.5 דקות ושאילתא 3 רצה 12.5 דקות. הזמן תלוי במספר הדפים בכל שאילתא וגם בכמה זמן לוקח לכל בקשה לגשת לדף להחזיר תשובה. בנוסף הוספנו בקוד שלנו השהייה(sleep) של שנייה לפני שמנסים לגשת לדף ולחלץ ממנו את התוכן כדי להקטין את הסיכוי שהזחלן שלנו ייחסם וזה גם משפיע על זמן הריצה. ניתן לשפר את זמן הריצה ע"י הקטנת הזמן של הsleep או לא להשתמש בsleep בכלל.

3. הזחלן החזיר לנו דפים שהם דפי קטגוריות מהם ניתן להגיע לקיצורים והחזיר לנו גם את הדפים של הקיצורים. אז בדפים שהוחזרו דפי הקטגוריות הם Hubs כי הם מפנים להרבה קישורים אחרים.

הדפים של הקיצורים הם authorities כי יותר נכנסים אליהם מדפים אחרים מאשר שהם מפנים לדפים אחרים בהשוואה לדפי קטגוריות- יכול להיות דרך דף של קיצור אחר(יש בתחתית הדפים האלה קישורים לערכים סמוכים) וגם דרך מספר דפי קטגוריות אם הקיצור שייך ליותר מקטגוריה אחת.

4.בחרנו 10 דפים של קיצורים משאילתא 3 שיש קשרים ביניהם. השתמשנו בקוד לחישוב הדירוג.

פלט ההרצה:

Initial PageRank Values:

PageRank values after iteration 0:

-----------------------------------

Page | PageRank Value

-----------------------------------

A | 0.10000

B | 0.10000

C | 0.10000

D | 0.10000

E | 0.10000

F | 0.10000

G | 0.10000

H | 0.10000

I | 0.10000

J | 0.10000

-----------------------------------

PageRank values after iteration 1:

-----------------------------------

Page | PageRank Value

-----------------------------------

A | 0.01500

B | 0.05750

C | 0.01500

D | 0.14250

E | 0.10000

F | 0.05750

G | 0.22750

H | 0.14250

I | 0.18500

J | 0.05750

-----------------------------------

PageRank values after iteration 2:

-----------------------------------

Page | PageRank Value

-----------------------------------

A | 0.01500

B | 0.02138

C | 0.01500

D | 0.07025

E | 0.13613

F | 0.02138

G | 0.15525

H | 0.21475

I | 0.25725

J | 0.09363

-----------------------------------

PageRank values after iteration 3:

-----------------------------------

Page | PageRank Value

-----------------------------------

A | 0.01500

B | 0.02138

C | 0.01500

D | 0.03954

E | 0.07471

F | 0.02138

G | 0.15525

H | 0.23010

I | 0.30331

J | 0.12433

-----------------------------------

PageRank values after iteration 4:

-----------------------------------

Page | PageRank Value

-----------------------------------

A | 0.01500

B | 0.02138

C | 0.01500

D | 0.03954

E | 0.04861

F | 0.02138

G | 0.10305

H | 0.26273

I | 0.32941

J | 0.14391

-----------------------------------

PageRank values after iteration 5:

-----------------------------------

Page | PageRank Value

-----------------------------------

A | 0.01500

B | 0.02138

C | 0.01500

D | 0.03954

E | 0.04861

F | 0.02138

G | 0.08086

H | 0.25996

I | 0.34328

J | 0.15500

-----------------------------------

**Highest PageRank after 5 iterations:**

**Page I with PageRank value of 0.34328**

**5.**

#### **משתמש 1:**

דפים רלוונטיים: דפים **H**, **I**, **J**.

דפים לא רלוונטיים: דפים **A**, **B**, **C**, **D**, **E**, **F**, **G**.

#### **משתמש 2:**

דפים רלוונטיים: דפים **I**, **H**, **G**, **J**.

דפים לא רלוונטיים: דפים **A**, **B**, **C**, **D**, **E**, **F**.

נבצע relevance feedback לפי המשוב של המשתמשים באופן הבא: נסתכל על איזה דפים המשתמשים הסכימו עליהם כרלוונטיים וניתן להם יותר משקל. נחשב דירוג חדש של הדפים עם התחשבות במשקלים האלה כך שתינתן עדיפות לדפים האלה.

שיפור שאילתא- ננסה להבין מה מאפיין את הדפים שסומנו כרלוונטיים לעומת לא רלוונטיים- למשל ייתכן כי לקיצור פחות מוכר/לא בשימוש תהיה הגדרה אחת בלבד ואז הוא כנראה יהיה פחות רלוונטי לשאילתא כי הוא מוריד את הממוצע של מס' הגדרות לקיצור. אז כדי לייעל את השאילתה נשנה את השאילתה כך שתסנן קיצורים כאלה ולא תיקח אותם בחשבון בחישוב.