# אחזור מידע: בניית מנוע חיפוש על ויקיפדיה

Querying Link to test our engine

קישור לקוד של הפרויקט כולל הוראות ברורות בקובץ README.md נמצאים בקישור הבא: קוד הפרויקט.

קישור ל- Google Storage Bucket המכיל את כל הדאטה והאינדקסים, נמצא בקישור הבא: Google Storage Bucket.

## מבוא:

הפרויקט שלנו יעסוק בבניית מנוע חיפוש לכל הקורפוס של English Wikipedia (מעל 6 מיליון מסמכים). מטרת הפרויקט היא יצירת מנוע חיפוש המקבל שאילתות כקלט ומחזיר כפלט את המסמכים הרלוונטיים ביותר עבורן (ע"פ מספר מדדים) בזמן הקצר ביותר.

## ניסויי מפתח ואופן הפעולה:

ראשית, נתאר את הדאטה שאיתו אנו עובדים. נגדיר כל מסמך בויקיפדיה על ידי 3 מאפיינים:
**Title** - כותרת העמוד, **Body** - הטקסט שמתאר את העמוד, **Anchor text** - לינקים מעמוד אל עמודים אחרים בויקיפדיה. הפרויקט מתעסק עם Big Data (גודל הקורפוס הוא מעל 6 מיליון מסמכים) ולכן החלטנו ליצור 3 Inverted Indices, אחד עבור ה- Title, אחד עבור ה- Body ואחד עבור ה- Anchor text.
בפונקציית search שהיא הפונקציה להחזרת התוצאות הטובות ביותר עבור שאילתה, בחרנו להתחשב במספר פרמטרים. ערכנו ניסויים ובדקנו מספר קומבינציות כדי למצוא את הערכים האופטימליים לפרמטרים. הפרמטרים שהתחשבנו בהם כדי להחזיר את התוצאות הרלוונטיות ביותר בזמן הקצר ביותר הם:

- אורך שאילתה: בדקנו האם עדיף לבצע חלוקה למקרים ולאחזר באמצעות שיטות שונות עבור שאילתות באורך שונה. הסקנו כי עבור שאילתות קצרות (שמכילות לכל היותר 3 מילים ייחודיות) יש לתת משקל משמעותי לחיפוש ע"פ כותרת.
- כמות מסמכים להחזיר: ניסינו בכל פעם להגביל את מספר המסמכים הרלוונטיים שיחזרו עבור כל שאילתה ע"י בדיקה של מספרים שונים. גילינו שלהחזיר 30 מסמכים או 40 מסמכים עבור סט האימון יניבו תוצאות זהות של ערכי MAP@40. לאחר בדיקה החלטנו להחזיר תוצאות של 5 המסמכים הרלוונטיים ביותר, מכיוון שעבורם קיבלנו את ערך MAP@40 המקסימלי. חשוב לציין שכל פונקציית חיפוש שבה אנו משתמשים מחזירה 60 תוצאות ע"מ לא לפספס מועמדים רלוונטיים לטופ 5 מסמכים.
- משקולות: בחרנו לתת משקולות לתוצאות של כל מדד (תוצאות עבור ה- body, title, anchor, page rank, page views). מדדים שנתנו תוצאות טובות ע"פ הניסויים שערכנו, יקבלו משקל גבוה יותר.

למשל: כאשר הרצנו רק את search_anchor() קיבלנו תוצאות רעות ולכן החלטנו לתת ל- anchor משקל 0 ולא להתחשב בו בהחזרת התוצאות הסופיות של המנוע. כלומר, ה- anchor היה מיותר עבורנו בהחזרת התוצאות אך הוא תרם בחישוב ה- Page Rank ע"מ למצוא דפים שמצביעים על דפים אחרים.

- Stopwords: לאחר מעבר על ה- train set וביצוע בדיקות, החלטנו להוסיף את המילה 'make' אל רשימת ה- stopwords מאחר וראינו כי באופן יחסי לשאילתות שקיבלנו הרבה מהן הכילו מילה זו (למדנו בקורס כי ניתן לעשות התאמות עבור סוג שאילתות מסוים שחוזר באופן תדיר) והיא לא תרמה לטעמנו הרבה למשמעות השאילתה, בנוסף שאילתות אלו קיבלו ציונים נמוכים. לאור הוספתה ל- stopwords ניכר שיפור דיי משמעותי במדד ה- Map@40, זה אמנם יוצר סיכון ל- Overfitting מאחר ואין לדעת מה יהיה ב- test set אך בחרנו לבצע זאת עקב המופע התדיר של המילה.

## אופטימיזציות:

כדי לבנות מנוע חיפוש אופטימלי ומספק עבור המשתמש, התחשבנו בשני המדדים החשובים ביותר: זמן להחזרת תוצאות, ודיוק המנוע (החזרת תוצאות רלוונטיות ככל שניתן).

**זמן להחזרת תוצאות**: כדי לצמצם את זמן הריצה מקבלת השאילתה ועד להחזרת המסמכים הרלוונטיים למשתמש, ביצענו חישובים מקדימים שמתבצעים ב- offline אשר מסייעים להחזיר תוצאות באופן מהיר יותר. המילונים מועלים לאוויר לפני ביצוע שאילתה של משתמש.

יצרנו מראש מילונים גלובליים אשר נטענים בפונקציה load_global(), בה גם ביצענו קריאה לאינדקסים שיצרנו. המילונים שיצרנו הם:

**IDF**

{term: idf}

מחזיק עבור כל מילה את ערך ה- idf שלה. נועד לגישה מהירה לערכי idf של כל מילה עבור חישוב ה- Cosine Similarity. יצרנו עבור body ו- title.

**Id title**

{doc_id: title}

מחזיק עבור כל מסמך את הכותרת שלו. נועד לגישה מהירה לכותרות ב- tuples שנחזיר בפונקציות ה- search השונות.

**DL**

{doc_id: doc length}

מחזיק עבור כל מסמך את האורך שלו. יצרנו מילון זה עבור ה- body (אורך הטקסט) ועבור ה- title (אורך הכותרת).

**Page Views**

{doc_id: views}

מחזיק עבור כל מסמך את כמות הצפיות בו בחודש אוגוסט 2021. נועד לגישה מהירה לצפיות במסמך והתחשבות בהם עבור פונקציית המשקל הכללית ב- search.

**Page Rank**

{doc_id: rank}

מחזיק עבור כל מסמך את הדירוג שלו. נועד לגישה מהירה לדירוג מסמך והתחשבות בדירוג עבור פונקציית המשקל הכללית ב- search.

**NORM**

{doc_id: norm}

מחזיק עבור כל מסמך את הנורמה שלו. מחושב בעזרת ה- tf – idf של כל מילה במסמך, נועד לגישה מהירה עבור חישוב ה- Cosine Similarity. יצרנו עבור body ו- title.

בנוסף, שמרנו את ה- bins בדיסק של ה- Instance במקום לקרוא אותם מה- bucket כדי לחסוך פעולות של קריאה מה- bucket ולתעדף קריאה מה- Local Disk.

**דיוק המנוע**: כדי להחזיר תוצאות מדויקות ככל שניתן בחרנו להתחשב בתוצאות שקיבלנו מה- Body וה- Title, תוך התחשבות בערכי ה- Page Rank ו- Page Views של המסמכים שחזרו. עבור ה- Body וה- Title בחרנו להשתמש ב- Cosine Similarity באמצעות tf-idf ועבור שאילתות המכילות לא יותר מ- 3 מילים ייחודיות בחרנו להשתמש רק ב- Cosine Similarity עבור ה- Title בשילוב עם Page Rank ו- Page Views.

בנוסף למילונים שהצגנו, יצרנו ב- offline גם מילונים שמחזיקים ערכים מנורמלים של Page Views, Rank כדי לחסוך בזמן ריצה וכדי לדייק בתוצאות של פונקציית search. פונקציה זו משקללת את ערכי ה- Cosine שחזרו מה- body, title, anchor (ערכים בתחום [-1,1]) עם ערכי ה- Page Views, Rank אך מכיוון שדירוג המסמך וכמות הצפיות אינם באותו סדר גודל החלטנו לבצע נרמול MinMax לערכים כדי למנוע הסטה של הציונים.

## שיטות הערכה למנוע, וממצאים:

לאחר כלל הניסויים שערכנו, הגענו למסקנה שכדי להגיע לתוצאות האופטימליות עבור מנוע החיפוש שלנו (תוך התחשבות ב- trade-off שבין דיוק התוצאות לבין זמן מענה על שאילתה), יש:

- ❖ לפעול בשיטה שונה בהתאם לאורך השאילתה. גילינו כי עבור שאילתות קצרות המכילות לכל היותר 3 מילים ייחודית, עדיף לאחזר את התוצאות שלהן ללא ה- Body ובכך לתת משקל משמעותי יותר ל- Title.
- ❖ להחזיר 5 מסמכים רלוונטיים בלבד כדי להקטין את הסיכוי להחזיר למשתמש תוצאות שהן לא רלוונטיות, זה אמנם סיכון לפספוס של רלוונטיים אך עזר מאוד להגדיל את ערך מדד MAP@40.
- ❖ לאחר בדיקות מעמיקות קיבענו את המשקולות של 0.3 ל- Body, 0.5 ל- Title, 0.1 ל- Page Rank ו- 0.1 ל- Page Views עבור שאילתות שאורכן גדול מ- 3 מילים ייחודיות ולעומת זאת: 0.1, 0.1,0,0.8,0 בהתאמה כאשר השאילתה מכילה לכל היותר 3 מילים ייחודיות.

עבור ערכי הפרמטרים הללו הצלחנו לקבל MAP@40 = 0.6466 וזמן ממוצע של 0.1836 שניות להחזרת תוצאות לשאילתה.

```
avg score: 0.6466333333333334 avg time: 0.1836044708887736
```

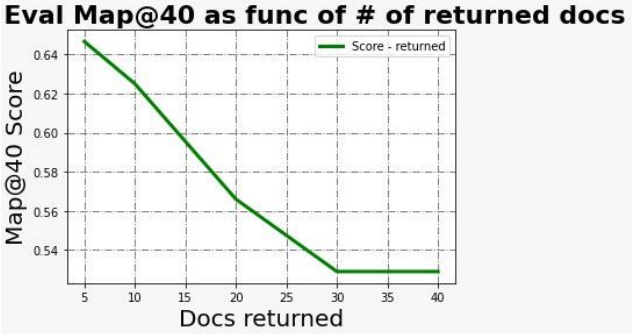
- הניסויים שערכנו ופירוט הביצועים של מנוע החיפוש עבור מדד MAP@40 וזמני ריצה ממוצעים מפורטים בסוף הדו"ח תחת 'נספחים'. מצורפות תמונות של התוצאות, כולל השוואות וקטעי קוד לגרפים.

## גרפים לביצועי מנוע החיפוש:

נציג גרפים הממחישים את ביצועי מנוע החיפוש שלנו ביחס לגרסאות מימוש שונות ולערכי פרמטרים שונים.

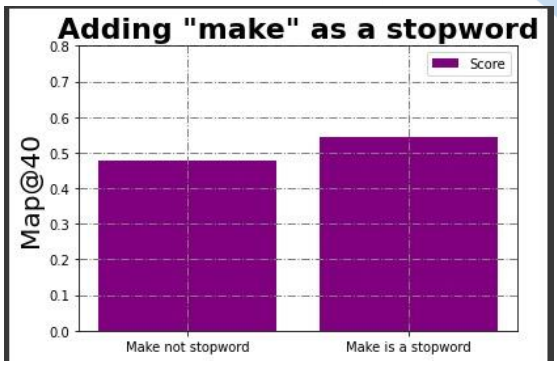
- גרף המציג את ביצועי המנוע לפי מדד MAP@40, עבור גרסאות שונות של מימוש.

כמות מסמכים שנחזיר: בדיקה של הביצועים כאשר נחזיר תוצאות של [40, 30, 20, 10, 5] המסמכים הרלוונטיים ביותר שמצא מנוע החיפוש.



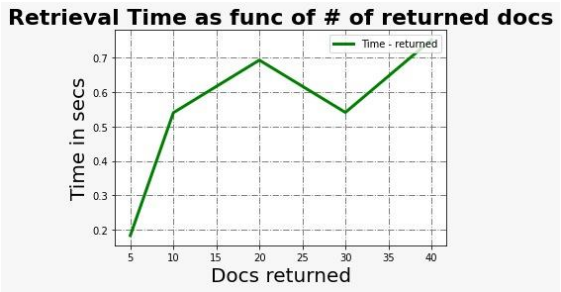

בהתבוננות בנוסחת המדד Map@40 זיהינו כי אם רוב המסמכים הראשונים שחוזרים הם רלוונטיים, אז משתלם להחזיר כמות תוצאות קטנה, ולכן ערכנו בדיקה של תלות המדד בכמות המסמכים המאוחזרים. אמנם החזרת חמישה מסמכים בלבד היא סיכון לפספוס של מסמכים רלוונטיים אך ראינו שברוב המוחלט של השאילתות כמות מסמכים זו עושה את העבודה.

StopWords: השפעת הוספת המילה ‘make’ לרשימת ה- Stopwords.




החלטנו להוסיף את המילה ‘make’ אל רשימת ה- stopwords מאחר וראינו כי באופן יחסי לשאילתות שקיבלנו הרבה מהן הכילו מילה זו והיא לא תרמה לטעמנו הרבה למשמעות השאילתה, בנוסף שאילתות אלו קיבלו ציונים נמוכים. לאור הוספתה ל- stopwords ניכר שיפור דיי משמעותי במדד ה- Map@40. הייתה עלייה של כ 0.05 בציון מה שלטעמנו משמעותי עבור מילה בודדת.

- גרף המציג את הזמן הממוצע לאחזור של שאילתה במנוע, עבור גרסאות שונות של מימוש.




באופן הגיוני, עקב החזרת כמות תשובות קטנה יותר ישנו שיפור בזמן הריצה. השיפור מאוד מינורי והינו בסדר גודל מאוד קטן, מה שגם גרם לירידה בזמן למרות אחזור יותר מסמכים (עבור 20 ו- 30 מסמכים).

הערכה איכותנית של 10 התוצאות המובילות עבור שאילתה אחת שבה המנוע שלנו תיפקד טוב מאוד ושאילתה אחת שבה המנוע לא הצליח:

```
[
  9146,
  "Dolly (sheep)"
],
[
  192685,
  "Dolly"
],
[
  4997271,
  "Dolly, Dolly, Dolly"
],
[
  17158563,
  "Sheep"
],
[
  21544233,
  "The Sheep"
],
[
  55366185,
  "Sheep (EP)"
],
[
  5003287,
  "Dolly (album)"
],
[
  65324501,
  "Dolly (song)"
],
[
  6922658,
  "Dolly James"
],
[
  20672385,
  "For the Love of Dolly"
]
]
```

עבור השאילתה ‘dolly the sheep’ חזר לנו ציון Map@40 של 1.0, כלומר הציון המקסימלי האפשרי ולהלן 10 התוצאות הראשונות שחזרו מהמנוע. אנו מאמינים כי בוצעה עבודה טובה של המנוע מאחר והגדרנו כי שאילתות מאורך קצר מ4 מילים ידורגו רק על סמך הכותרת, הצפיות והדירוגים. על סמך הבדיקות שערכנו אינדקס הכותרות על סמך דמיון cosine מספק ביצועים מצוינים והוא הוכיח את עצמו גם במקרה זה.

עבור השאילתה ‘how kids come to world?’ חזר לנו ציון Map@40 של 0. אנו מאמינים שזה קרה מאחר וכאשר כל מילה בשאילתה עומדת בפני עצמה יש לה משמעות שונה מאוד מצירוף המילים יחד והרבה תוצאות לא רלוונטיות חזרו עבורן. מאחר ולצערנו לא מימשנו מודלים המתחשבים בקונספטים כמו LSI ו/או אינדקס לצירופי זוגות, פספסנו את משמעות השאילתה ולא הצלחנו "לתפוס" מסמכים רלוונטיים.

```
  5105656,
  "Kids World"
],
[
  17509922,
  "World of Kids"
],
[
  1355560,
  "Come into My World"
],
[
  8163829,
  "World to come"
],
[
  32529911,
  "Come into the World"
],
[
  49198612,
  "A World Not to Come"
],
[
  57549235,
  "Come into Our World"
],
[
  59884820,
  "The World to Come"
],
[
  42072639,
  "Kids World (film)"
],
[
  481171,
  "Kids' WB"
]
```

- בעמוד למטה נשמח להפנות אל תיאור ההגבלות בפרויקט ותכניות לעבודה עתידית שלנו כחלק מהנספחים של הפרויקט.

## הגבלות ועבודה עתידית:

במידה והיו לנו עוד מספר חודשים של מחקר, היינו מתעמקים יותר בנושא כדי להגיע למסקנות ובכך לבנות מנוע חיפוש שמחזיר תוצאות מדויקות יותר ובזמנים טובים יותר. שיטות שהיינו חוקרים במידה והיה לנו זמן נוסף:

1. יצירת מילון שמכיל עבור כל מילה את כמות המופעים הכוללת שלה בקורפוס (term total). בעזרת מילון זה ובהסתמך על Zipf's Law, נוכל להוסיף את המילים השכיחות ביותר והנדירות ביותר בקורפוס כ- Stopwords.

2. יצירת אינדקס ל- bi-grams אשר יכול לשפר את דיוק התוצאות. bi-grams לוכדות את ההקשר והמשמעות של המילים בטקסט, שיכולים ללכת לאיבוד כאשר משתמשים רק במילים בודדות.

בכוונתנו להמשיך לחקור את הנושא, לממש עוד שיטות ולהעמיק בשיטות נוספות שיכולות לסייע לנו בשיפור מנוע החיפוש שבנינו. הפרויקט העשיר אותנו בתחום אחזור המידע ואנו מאמינים כי נוכל לשפר את מנוע החיפוש שלנו ולחקור עוד שיטות מסקרנות.

## נספחים:

תוצאות של הרצת סט האימון על מנוע החיפוש, ע"פ מדד MAP@40 וזמן ריצה ממוצע לשאילתה.

שינוי בכמות המסמכים שבחרנו להחזיר:

כמות מסמכים להחזיר: 5.

ציון MAP@40: 0.6466. זמן ריצה ממוצע לשאילתה: 0.1836 שניות.

```
[12]
[('best marvel movie', 0.162736177444458, 0.0),
 ('How do kids come to world?', 0.6444768905639648, 0.0),
 ('Information retrieval', 0.07696175575256348, 1.0),
 ('LinkedIn', 0.05324149131774902, 1.0),
 ('How to make coffee?', 0.05954432487487793, 0.75),
 ('Ritalin', 0.05352950096130371, 1.0),
 ('How to make wine at home?', 0.10507678985595703, 0.75),
 ('Most expensive city in the world', 0.6409549713134766, 0.333),
 ('India', 0.14835619926452637, 1.0),
 ('how to make money fast?', 0.07404804229736328, 0.417),
 ('Netflix', 0.05427145957946777, 0.887),
 ('Apple computer', 0.07888936996459961, 0.7),
 ('The Simpsons', 0.05585789680480957, 0.917),
 ('World cup', 0.6849000453948975, 1.0),
 ('How to lose weight?', 0.058277130126953125, 0.0),
 ('Java', 0.05684018135070801, 0.5),
 ('Air Jordan', 0.16437983512878418, 1.0),
 ('how to deal with depression?', 0.05908703804016113, 0.5),
 ('How do you make gold', 0.08389496803283691, 1.0),
 ('Marijuana', 0.05302786827087402, 0.25),
 ('How to make hummus', 0.053598642349243164, 0.833),
 ('Winter', 0.11679220199584961, 0.806),
 ('Rick and Morty', 0.0685579776763916, 1.0),
 ('Natural Language processing', 0.17502307891845703, 1.0),
 ('World Cup 2022', 1.267235279083252, 1.0),
 ('Dolly the sheep', 0.06338214874267578, 1.0),
 ('Cigarettes', 0.05518388748168945, 0.0),
 ('What is the best place to live in?', 0.2236161231994629, 0.0),
 ('Elon musk', 0.0560657978057861 3, 0.756),
 ('How do you breed flowers?', 0.06032705307006836, 0.0)]

1 print("avg score: "+str(sum(scores)/30) + " avg time: "+ str(sum(times)/30))

avg score: 0.6466333333333334 avg time: 0.1836044708887736
```

כמות מסמכים להחזיר: 10.

ציון MAP@40: 0.625. זמן ריצה ממוצע לשאילתה: 0.5399 שניות.

```
[('best marvel movie', None, None),
 ('How do kids come to world?', 11.295539855957031, 0.0),
 ('Information retrieval', 0.09389972686767578, 0.957),
 ('LinkedIn', 0.0682671070098877, 1.0),
 ('How to make coffee?', 0.07455706596374512, 0.75),
 ('Ritalin', 0.06647944450378418, 1.0),
 ('How to make wine at home?', 0.12393331527709961, 0.75),
 ('Most expensive city in the world', 0.7386350631713867, 0.31),
 ('India', 0.12574267387390137, 1.0),
 ('how to make money fast?', 0.08668160438537598, 0.417),
 ('Netflix', 0.06861019134521484, 0.849),
 ('Apple computer', 0.09679150581359863, 0.7),
 ('The Simpsons', 0.06899213790893555, 0.917),
 ('World cup', 0.8152878284454346, 1.0),
 ('How to lose weight?', 0.07169985771179199, 0.0),
 ('Java', 0.07266354560852051, 0.375),
 ('Air Jordan', 0.1832575798034668, 1.0),
 ('how to deal with depression?', 0.07426166534423828, 0.5),
 ('How do you make gold', 0.09929561614990234, 1.0),
 ('Marijuana', 0.06888365745544434, 0.225),
 ('How to make hummus', 0.06814694404602051, 0.833),
 ('Winter', 0.1310429573059082, 0.806),
 ('Rick and Morty', 0.08423018455505371, 1.0),
 ('Natural Language processing', 0.20252084732055664, 0.933),
 ('World Cup 2022', 0.7619760036468506, 0.507),
 ('Dolly the sheep', 0.07726407051086426, 1.0),
 ('Cigarettes', 0.07001566886901855, 0.167),
 ('What is the best place to live in?', 0.3607454299926758, 0.0),
 ('Elon musk', 0.07076001167297363, 0.756),
 ('How do you breed flowers?', 0.07762956619262695, 0.0)]

1 print("avg score: "+str(sum(scores)/30) + " avg time: "+ str(sum(times)/30))

avg score: 0.6250666666666668 avg time: 0.5399270375569661
```

כמות מסמכים להחזיר: 20.

ציון MAP@40: 0.5668. זמן ריצה ממוצע לשאילתה: 0.6938 שניות.

```
[('best marvel movie', None, None),
 ('How do kids come to world?', 15.98622441291809, 0.0),
 ('Information retrieval', 0.09156632423400879, 0.845),
 ('LinkedIn', 0.06737685203552246, 1.0),
 ('How to make coffee?', 0.07477211952209473, 0.75),
 ('Ritalin', 0.06767511367797852, 1.0),
 ('How to make wine at home?', 0.12170171737670898, 0.75),
 ('Most expensive city in the world', 0.7110874652862549, 0.31),
 ('India', 0.12550044059753418, 0.733),
 ('how to make money fast?', 0.08590960502624512, 0.328),
 ('Netflix', 0.0676872730255127, 0.77),
 ('Apple computer', 0.09400749206542969, 0.7),
 ('The Simpsons', 0.06891202926635742, 0.599),
 ('World cup', 0.8149540424346924, 1.0),
 ('How to lose weight?', 0.0727994441986084, 0.0),
 ('Java', 0.07201766967773438, 0.302),
 ('Air Jordan', 0.18326115608215332, 0.559),
 ('how to deal with depression?', 0.0743110179901123, 0.5),
 ('How do you make gold', 0.10071754455566406, 1.0),
 ('Marijuana', 0.06800532341003418, 0.215),
 ('How to make hummus', 0.06604337692260742, 0.833),
 ('Winter', 0.1311047077178955, 0.654),
 ('Rick and Morty', 0.08222079277038574, 0.98),
 ('Natural Language processing', 0.21071338653564453, 0.878),
 ('World Cup 2022', 0.7440154552459717, 0.507),
 ('Dolly the sheep', 0.07530045509338379, 1.0),
 ('Cigarettes', 0.06910467147827148, 0.167),
 ('What is the best place to live in?', 0.3442842960357666, 0.0),
 ('Elon musk', 0.07041072845458984, 0.625),
 ('How do you breed flowers?', 0.07488417625427246, 0.0)]

1 print("avg score: "+str(sum(scores)/30) + " avg time: "+ str(sum(times)/30))

avg score: 0.5668333333333334 avg time: 0.6938856363296508
```

כמות מסמכים להחזיר: 30 / 40 (התקבלו אותן תוצאות ב- MAP@40 וזמן הריצה עבור 40 מסמכים גדול יותר).

ציון MAP@40: 0.5296. זמן ריצה ממוצע לשאילתה: 0.5414 שניות.

ציון MAP@40: 0.5296. זמן ריצה ממוצע לשאילתה: 0.752 שניות.

```
[('best marvel movie', None, None),
 ('How do kids come to world?', 11.244789600372314, 0.0),
 ('Information retrieval', 0.09173870086669922, 0.81),
 ('LinkedIn', 0.06932282447814941, 1.0),
 ('How to make coffee?', 0.07572579383850098, 0.526),
 ('Ritalin', 0.06680107116699219, 1.0),
 ('How to make wine at home?', 0.12386059761047363, 0.75),
 ('Most expensive city in the world', 0.7489628791809082, 0.199),
 ('India', 0.12665915489196777, 0.509),
 ('how to make money fast?', 0.0888831615447998, 0.328),
 ('Netflix', 0.06958627700805664, 0.654),
 ('Apple computer', 0.09593725204467773, 0.7),
 ('The Simpsons', 0.07095217704772949, 0.54),
 ('World cup', 0.8510980606079102, 1.0),
 ('How to lose weight?', 0.07361769676208496, 0.0),
 ('Java', 0.07324099540710449, 0.298),
 ('Air Jordan', 0.18603205680847168, 0.559),
 ('how to deal with depression?', 0.07493138313293457, 0.5),
 ('How do you make gold', 0.10168862342834473, 1.0),
 ('Marijuana', 0.06833076477050781, 0.204),
 ('How to make hummus', 0.0674750804901123, 0.833),
 ('Winter', 0.13438653945922852, 0.654),
 ('Rick and Morty', 0.08452367782592773, 0.98),
 ('Natural Language processing', 0.22766542434692383, 0.878),
 ('World Cup 2022', 0.7688858509063721, 0.309),
 ('Dolly the sheep', 0.07756352424621582, 1.0),
 ('Cigarettes', 0.0671534538269043, 0.113),
 ('What is the best place to live in?', 0.3651461601257324, 0.0),
 ('Elon musk', 0.07090520858764648, 0.546),
 ('How do you breed flowers?', 0.07694625854492188, 0.0)]

[59]  1 print("avg score: "+str(sum(scores)/30) + " avg time: "+ str(sum(times)/30))

avg score: 0.5296666666666667 avg time: 0.5414270083109538
```

סיכום ביצועים ע"פ מדד MAP@40 בשינוי ערך הפרמטר 'כמות מסמכים להחזיר':

| #Docs | MAP@40 | Average run time |
|---|---|---|
| 5 | 0.6466 | 0.1836 |
| 10 | 0.625 | 0.5399 |
| 20 | 0.5668 | 0.6938 |
| 30 | 0.5296 | 0.5414 |
| 40 | 0.5296 | 0.752 |

קטעי קוד של יצירת הגרפים:

```
import matplotlib.pyplot as plt
x = [5, 10, 20, 30, 40]
y = [0.6466, 0.625, 0.566, 0.529, 0.529]
plt.plot(x, y, color='green', linewidth=3)
plt.xlabel('Docs returned' , fontsize=20)
plt.ylabel('Map@40 Score',fontsize=20)
plt.title('Engine Eval Map@40 as func of # of returned docs',fontsize=22,fontweight= 'bold')
plt.grid(color='grey', linewidth=1, linestyle='-.')
plt.legend(['Score - returned'],loc= 'upper right')
plt.show()
```

```
import matplotlib.pyplot as plt
x = [5, 10, 20, 30, 40]
y = [0.183, 0.54, 0.693, 0.541, 0.752]
plt.plot(x, y, color='green', linewidth=3)
plt.xlabel('Docs returned' , fontsize=20)
plt.ylabel('Time in secs',fontsize=20)
plt.title('Retrieval Time as func of # of returned docs',fontsize=22,fontweight= 'bold')
plt.grid(color='grey', linewidth=1, linestyle='-.')
plt.legend(['Time - returned'],loc= 'upper right')
plt.show()
```

רשימת כל קבצי האינדקסים:

מצורפים 5 צילומי מסך של הקבצים אשר נמצאים ב- Bucket.

תיקיית postings_gcp מכילה את קבצי ה- Body Inverted Index הכוללים את קבצי ה- pkl של האינדקס עצמו וה- bins.

תיקיית title_index מכילה את קבצי ה- Title Inverted Index הכוללים את קבצי ה- pkl של האינדקס עצמו וה- bins.

תיקיית anchor_index מכילה את קבצי ה- Anchor Inverted Index הכוללים את קבצי ה- pkl של האינדקס עצמו וה- bins.

תיקיית global stats מכילה את הסטטיסטיקות הגלובליות שלנו שהן קבצי ה- pkl של המילונים שבהן השתמשנו לחישובים.

```
1.91 MiB     gs://208906255/postings_gcp/9_008.bin
1.91 MiB     gs://208906255/postings_gcp/9_009.bin
1.91 MiB     gs://208906255/postings_gcp/9_010.bin
1.91 MiB     gs://208906255/postings_gcp/9_011.bin
1.91 MiB     gs://208906255/postings_gcp/9_012.bin
1.91 MiB     gs://208906255/postings_gcp/9_013.bin
1.91 MiB     gs://208906255/postings_gcp/9_014.bin
1.91 MiB     gs://208906255/postings_gcp/9_015.bin
1.91 MiB     gs://208906255/postings_gcp/9_016.bin
1.91 MiB     gs://208906255/postings_gcp/9_017.bin
1.91 MiB     gs://208906255/postings_gcp/9_018.bin
1.91 MiB     gs://208906255/postings_gcp/9_019.bin
1.91 MiB     gs://208906255/postings_gcp/9_020.bin
1.91 MiB     gs://208906255/postings_gcp/9_021.bin
1.26 MiB     gs://208906255/postings_gcp/9_022.bin
100.26 KiB   gs://208906255/postings_gcp/9_posting_locs.pickle
18.45 MiB    gs://208906255/postings_gcp/index.pkl
5.92 GiB     gs://208906255/postings_gcp/
0 B          gs://208906255/pr/
0 B          gs://208906255/pr/_SUCCESS
62.9 MiB     gs://208906255/pr/part-00000-581cb96f-799a-41f2-9c3d-e7e087c5e3ed-c000.csv.gz
62.9 MiB     gs://208906255/pr/pr.gz
125.79 MiB   gs://208906255/pr/
861.88 KiB   gs://208906255/title_index/248_000.bin
324.46 KiB   gs://208906255/title_index/248_posting_locs.pickle
710.26 KiB   gs://208906255/title_index/249_000.bin
325.99 KiB   gs://208906255/title_index/249_posting_locs.pickle
632.68 KiB   gs://208906255/title_index/250_000.bin
323.48 KiB   gs://208906255/title_index/250_posting_locs.pickle
```

```
0 B          gs://208906255/global_stats/
44.28 MiB    gs://208906255/global_stats/d2dl_dict.pkl
168.88 MiB   gs://208906255/global_stats/id2title_dict.pkl
84.73 MiB    gs://208906255/global_stats/norm2doc_dict.pkl
73.5 MiB     gs://208906255/global_stats/page_views.pkl
286.4 MiB    gs://208906255/global_stats/pageviews-202108-user-4dedup.txt
42.29 MiB    gs://208906255/global_stats/title_d2dl_dict.pkl
84.73 MiB    gs://208906255/global_stats/title_norm2doc_dict.pkl
34.81 MiB    gs://208906255/global_stats/title_w2idf_dict.pkl
6.22 MiB     gs://208906255/global_stats/w2df_dict.pkl
9.37 MiB     gs://208906255/global_stats/w2idf_dict.pkl
835.2 MiB    gs://208906255/global_stats/
1.91 MiB     gs://208906255/postings_gcp/0_000.bin
1.91 MiB     gs://208906255/postings_gcp/0_001.bin
1.91 MiB     gs://208906255/postings_gcp/0_002.bin
1.91 MiB     gs://208906255/postings_gcp/0_003.bin
1.91 MiB     gs://208906255/postings_gcp/0_004.bin
1.91 MiB     gs://208906255/postings_gcp/0_005.bin
1.91 MiB     gs://208906255/postings_gcp/0_006.bin
1.91 MiB     gs://208906255/postings_gcp/0_007.bin
1.91 MiB     gs://208906255/postings_gcp/0_008.bin
1.91 MiB     gs://208906255/postings_gcp/0_009.bin
```

```
1.91 MiB     gs://208906255/anchor_index/124_000.bin
1.91 MiB     gs://208906255/anchor_index/124_001.bin
1.91 MiB     gs://208906255/anchor_index/124_002.bin
1.91 MiB     gs://208906255/anchor_index/124_003.bin
1.91 MiB     gs://208906255/anchor_index/124_004.bin
1.91 MiB     gs://208906255/anchor_index/124_005.bin
1.91 MiB     gs://208906255/anchor_index/124_006.bin
1.91 MiB     gs://208906255/anchor_index/124_007.bin
1.91 MiB     gs://208906255/anchor_index/124_008.bin
1.91 MiB     gs://208906255/anchor_index/124_009.bin
1.91 MiB     gs://208906255/anchor_index/124_010.bin
1.08 MiB     gs://208906255/anchor_index/124_011.bin
509.57 KiB   gs://208906255/anchor_index/124_posting_locs.pickle
1.91 MiB     gs://208906255/anchor_index/125_000.bin
1.91 MiB     gs://208906255/anchor_index/125_001.bin
1.91 MiB     gs://208906255/anchor_index/125_002.bin
1.91 MiB     gs://208906255/anchor_index/125_003.bin
1.91 MiB     gs://208906255/anchor_index/125_004.bin
1.91 MiB     gs://208906255/anchor_index/125_005.bin
1.91 MiB     gs://208906255/anchor_index/125_006.bin
1.91 MiB     gs://208906255/anchor_index/125_007.bin
1.91 MiB     gs://208906255/anchor_index/125_008.bin
1.27 MiB     gs://208906255/anchor_index/125_009.bin
513.13 KiB   gs://208906255/anchor_index/125_posting_locs.pickle
1.91 MiB     gs://208906255/anchor_index/126_000.bin
1.91 MiB     gs://208906255/anchor_index/126_001.bin
1.91 MiB     gs://208906255/anchor_index/126_002.bin
```

```
239.43 MiB   gs://208906255/multistream21_part2_preprocessed.parquet
212.83 MiB   gs://208906255/multistream21_part3_preprocessed.parquet
211.5 MiB    gs://208906255/multistream21_preprocessed.parquet
247.25 MiB   gs://208906255/multistream22_part2_preprocessed.parquet
210.15 MiB   gs://208906255/multistream22_part3_preprocessed.parquet
41.59 MiB    gs://208906255/multistream22_part4_preprocessed.parquet
204.39 MiB   gs://208906255/multistream22_preprocessed.parquet
252.62 MiB   gs://208906255/multistream23_part2_preprocessed.parquet
178.21 MiB   gs://208906255/multistream23_part3_preprocessed.parquet
160.49 MiB   gs://208906255/multistream23_part4_preprocessed.parquet
135.36 MiB   gs://208906255/multistream23_preprocessed.parquet
212.18 MiB   gs://208906255/multistream24_part2_preprocessed.parquet
171.82 MiB   gs://208906255/multistream24_part3_preprocessed.parquet
212.79 MiB   gs://208906255/multistream24_part4_preprocessed.parquet
66.56 MiB    gs://208906255/multistream24_part5_preprocessed.parquet
181.21 MiB   gs://208906255/multistream24_preprocessed.parquet
187.69 MiB   gs://208906255/multistream25_part2_preprocessed.parquet
168.84 MiB   gs://208906255/multistream25_part3_preprocessed.parquet
134.23 MiB   gs://208906255/multistream25_part4_preprocessed.parquet
171.28 MiB   gs://208906255/multistream25_preprocessed.parquet
163.67 MiB   gs://208906255/multistream26_preprocessed.parquet
182.28 MiB   gs://208906255/multistream27_part2_preprocessed.parquet
163.03 MiB   gs://208906255/multistream27_part3_preprocessed.parquet
146.99 MiB   gs://208906255/multistream27_preprocessed.parquet
429.02 MiB   gs://208906255/multistream2_preprocessed.parquet
461.92 MiB   gs://208906255/multistream3_preprocessed.parquet
495.78 MiB   gs://208906255/multistream4_preprocessed.parquet
514.62 MiB   gs://208906255/multistream5_preprocessed.parquet
538 MiB      gs://208906255/multistream6_preprocessed.parquet
```

```
328.11 KiB   gs://208906255/title_index/354_posting_locs.pickle
1.06 MiB     gs://208906255/title_index/355_000.bin
330.17 KiB   gs://208906255/title_index/355_posting_locs.pickle
540.16 KiB   gs://208906255/title_index/356_000.bin
327.29 KiB   gs://208906255/title_index/356_posting_locs.pickle
593.54 KiB   gs://208906255/title_index/357_000.bin
330.04 KiB   gs://208906255/title_index/357_posting_locs.pickle
1 MiB        gs://208906255/title_index/358_000.bin
326.3 KiB    gs://208906255/title_index/358_posting_locs.pickle
686.68 KiB   gs://208906255/title_index/359_000.bin
324.39 KiB   gs://208906255/title_index/359_posting_locs.pickle
919.52 KiB   gs://208906255/title_index/360_000.bin
328.82 KiB   gs://208906255/title_index/360_posting_locs.pickle
643.37 KiB   gs://208906255/title_index/361_000.bin
323.9 KiB    gs://208906255/title_index/361_posting_locs.pickle
534.25 KiB   gs://208906255/title_index/362_000.bin
329.36 KiB   gs://208906255/title_index/362_posting_locs.pickle
545.72 KiB   gs://208906255/title_index/363_000.bin
321.31 KiB   gs://208906255/title_index/363_posting_locs.pickle
643.6 KiB    gs://208906255/title_index/364_000.bin
```