205915135 רם מנור ramman@post.bgu.ac.il
208836247 אדר הררי adarha@post.bgu.ac.il
לינק Github : Guthub
לינק Gcp_bucket: Google Storage Bucket_
IP למכונה : 104.198.17.122

דו"ח פרויקט בקורס אחזור מידע:

בתחילת העבודה על הפרויקט עבדנו תחילה עם מחברת colab על מספר קטן של מסמכים ולאחר שהבנו איך להמשיך יצרנו דרך הgcp את שלושת האינדקסים של הקורפוס – title, ה anchor, body . בנוסף אליהם יצרנו מספר קבצי pickle :

מילון השומר לכל id את הכותרת שלו, מילון השומר לכל מסמך את דירוג הpage rank שלו , מילון השומר לכל מסמך את כמות הצפיות שלו. מילון DL השומר את כמות המילים פר מסמך לאחר סינון מילים.

לאחר בניית האינדקסים על הקורפוס המלא התחלנו לבצע את המתודות של הפונקציות לחיפוש כפי שהוגדרו לנו בקובץ.

עם הפונקציות האלו ניסינו ליצור פונקציית חיפוש טובה אשר משקללת גם את התוצאות את פונקציית חיפוש הכותרת וגם את פונקציית החיפוש של הטקסט.

הרצנו כל פונקציית חיפוש בנפרד ושמנו לב שכאשר יש בשאילתה מילה 1 או 2 פונקציית חיפוש של הכותרת עבדה יותר טוב משל הטקסט. ולכן שקלנו לבדוק את אורך השאילתה לאחר סינון ואם יתקבל אורך קטן מ2 או שווה ל1, נפנה את החיפוש לפונקציית חיפוש הכותרות.

לדוגמא :

```
('Information retrieval', 0.021145105361938477, 0.89), ('LinkedIn', 0.004002809524536133, 1.0),
```

```
('Ritalin', 0.0027370452880859375, 1.0),
```

ואילו של הטקסט:

```
('Information retrieval', 3.9253249168395996, 0.0),
('Ritalin', 0.5169651508331299, 0.557)('LinkedIn', 0.052840232849121094, 0.0),
```

ראינו כי פונקציית חיפוש הכותרות לא עובדת טוב עם שאילתות ארוכות ולכן נצטרך לשלב עוד חיפושים. הסיבה לכך היא ששאילתות שמורכבות ממספר מילים בודדות מכילות כנראה מילות מפתח שאותן קל למצוא בחיפוש בכותרות אשר גם הן מכילות לרוב מילות מפתח. ובשאילתות ארוכות יותר קשה לה למצוא התאמות- יש פחות מילות מפתח ויש לעשות חיפוש אחר כדי להתאים את השאילתה למסמכים.

לכן, החלטנו גם לנסות לשלב משקלים בין שתי פונקציות החיפוש של הטקסט ושל הכותרות.

דוגמא לאחת מפונקציות החיפוש שניסינו - הרצנו על השאילתות את פונקציית חיפוש הכותרת בנפרד ולקחנו את ה200 תוצאות ראשונות ואת פונקציית חיפוש הטקסט בנפרד.

הגדרנו משקלים לכותרת (0.1) וטקסט (0.9). יצרנו פונקציית חיפוש המתייחסת למיקום של כל תוצאה גם לפי הטקסט וגם לפי הכותרות והוספנו שכלול של התוצאות מפונקציית החיפוש של anchor אך ראינו כי השילוב של anchor פוגע לנו במדדים ולכן החלטנו לא להשתמש בו. בפונקציה זו הכפלנו את מיקום המסמך שהוחזר לפי החיפושים הכלליים במשקל שהגדרנו ואיחדנו בין התוצאות. כאשר דירוג של מסמך שחזר קודם קיבל דירוג נמוך יותר גם לפי הכותרת וגם לפי הטקסט. ככל שדירוג גבוה יותר כך המסמך פחות רלוונטי (כי המיקום שלו לא במקומות הראשונים). אך המדדים לא היו טובים.

יצרנו פונקציה נוספת שמחפשת את השאילתה גם בפונקציית חיפוש הכותרות וגם בפונקציית חיפוש הטקסט ואיחדנו בין התוצאות, בדקנו בתוצאות שחזרו את המילים שחוזרות הכי הרבה ואותן צירפנו לשאילתה ושלחנו את השאילתה המקורית בתוספת של מילים נפוצות לפונקציית חיפוש של הטקסט. חשבנו כי זה ישפר לנו את התוצאות אך התוצאות היו מאוד לא טובות וזמני ריצה ארוכים מאוד .

הפונקציה שיצאה הכי גבוהה עד כה מבחינת התוצאה הייתה של פונקציית חיפוש הכותרות אך לא יכולנו להסתמך על כך ולקחת אותה לפונקציית החיפוש שלנו - אנחנו יכולים להגיע למצב של overfitting . אם לדוגמא נסתמך רק על פונקציית חיפוש הכותרות אך בפועל נקבל שאילתות אשר מכילות יותר מ3 מילים כנראה שלא נמצא התאמה טובה לשאילתה. ולכן בחרנו להשתמש ב**BM 25** במקום פונקציית החיפוש של הטקסט (שהביאה תוצאה של שקרובה מאוד לאפס).**BM 25** לוקח בחשבון מספר גורמים המשפיעים על הרלוונטיות, כגון תדירות מונחי השאילתה במסמך והאורך הכולל של המסמך. ראינו כי אם משתמשים רק בה התוצאות משתפרות. בנוסף יצרנו פונקציה שמפנה את החיפוש לפונקציית חיפוש הכותרות אם האורך קטן מ-1 לבדוק האם זה שיפר את התוצאה. בניסויים שערכנו ראינו כי זה לא חד משמעי, לפעמים זה משפר את החיפוש ולפעמים לא ולכן לא השתמשנו ברעיון זה.

באחד הניסיונות הוספנו שינוי בפונקציית **BM 25** שתנסה לשפר את התוצאות שלנו על ידי בחירה אקראית של מילה מהשאילתה וחיפוש שלה ספציפית באינדקס הכותרת וכן בגוף האינדקס ונתינת משקל גבוה יותר בגמר החישוב לאותם מסמכים שלפחות פעם אחת מופיע המילה של השאילתה גם בbody וגם ב title . אך בגלל האקראיות קיבלנו תוצאות שונות ולא יכלנו להסתמך עליה.

החלטנו ללכת על **BM 25** כאשר אנו עוברים רק על המסמכים שהמילה מה- **QUERY** שהתקבלה מופיעה במסמך למעלה מ 50 פעמים. זאת מההנחה שמסמך אשר יכיל תדירות גבוהה של המילים בתוכן של המאמר יהיה יותר רלוונטי מכאלה שיש להם פחות .
לפיכך קבענו Threshold (50) על מנת לתעדף את אותם מסמכים. ואכן זאת הפונקציה שהביאה לנו את המדד הכי גבוה וזמני ריצה טובים.

להלן המדדים של פונקציית החיפוש עבור 10 השאילתות של הקובץ:

Topic	Score
How to make hummus	1.000
Winter	0.318
Rick and Morty	0.818
Natural Language processing	1.000
World Cup 2022	0.000
Dolly the sheep	0.131
Cigarettes	0.643
What is the best place to live in?	0.000
Elon musk	0.887
How do you breed flowers?	0.092

ה40@MAP הינו – 0.488

אלו התוצאות הטובות עבור השאילתה Nutural Language processing מתוך פונקציית החיפוש שלנו:

```
Outline of natural language processing
USCIS processing times
Heuristic-systematic model of information processing
Transaction processing system
Premium Processing Service
Central place foraging
Food processing
Array processing
Auditory processing disorder
Stream processing
```

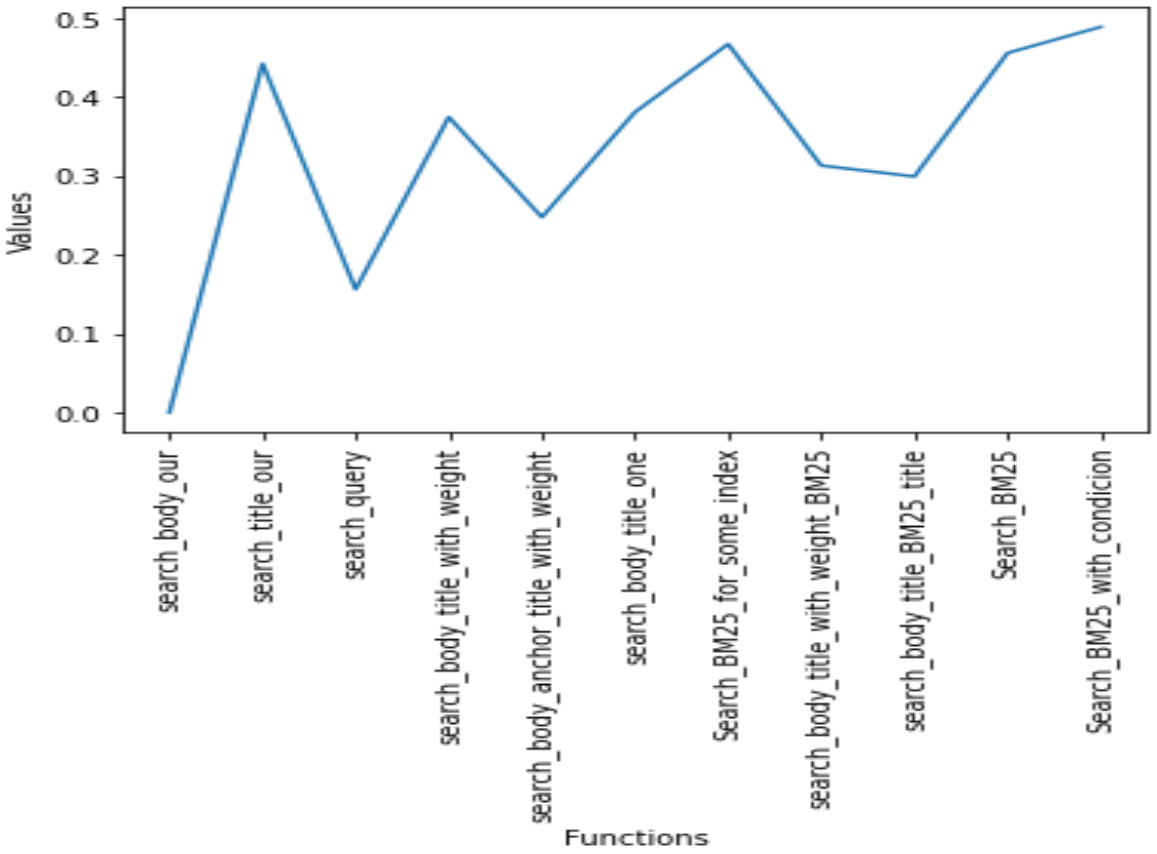
אלו התוצאות הלא טובות עבור השאילתה - what is the best place to live in? מתוך פונקציית החיפוש שלנו:

```
Uma Canção para Ti (Portugal)
List of awards and nominations received by Armin van Buuren
UK Rock Challenge
List of Japanese adult video awards (1991–2008)
University of San Agustin Publications
List of American theatrical animated feature films (2000-2019)
List of The Best Show with Tom Scharpling episodes
List of albums titled Live
Toronto Rock 'n' Roll Revival 1969, Volume IV
List of The X Factor (British TV series) episodes
```

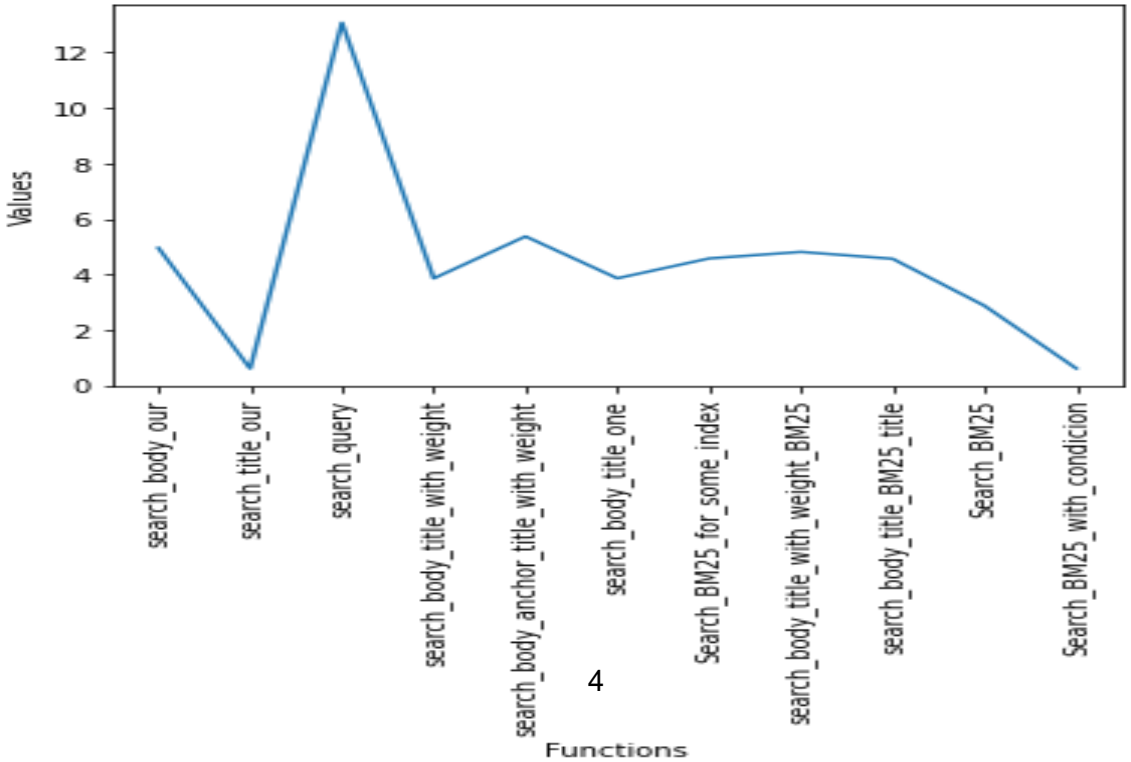
באופן כללי ניתן לראות כי לפונקציית החיפוש שלנו קשה להתמודד עם שאילתות ארוכות ואילו לשאילתות קצרות מצאה תוצאות טובות לרוב. כנראה כי לא היה ברור אילו מילים

חשובות יותר ואילו לא. ואילו בשאילתות קצרות כנראה שכל מילה חשובה ויותר קלה לה למצוא התאמות לכך. אם היה לנו עוד זמן לפרויקט היינו מנסים להשתמש במודלים של למידה עמוקה ומבצעים stremming למיקסום החיפוש שלנו.

גרף המתאר את הביצועים של הגרסאות שלנו עבור 10 השאילתות מתוך ה test שלנו:



גרף המתאר את ביצועי הזמן של הגרסאות שלנו עבור 10 השאילתות מתוך ה test שלנו:



נספח – הסבר לפי שמות הפונקציות מהגרפים:

search_body_our – פונקציה של חיפוש בטקסט המסמך לפי tfidf וcosine similarity.

search_title_our - פונקציית חיפוש בכותרות המסמך כפי שהוגדר בקובץ.

search_query – מחפש מילים נפוצות מתוך המסמכים שהוחזרו מפונקציית הכותרות ומפונקציית הטקסט וחיפוש מחדש בפונקציית הטקסט.

search_body_title_with_weight – חיפוש השאילתה בפונקציית הכותרות ופונקציית החיפוש בנפרד ומיזוג התוצאות ע"י סדר חזרתם מהפונקציות (עם משקלים).

search_body_anchor_title_with_weight – אותו הדבר כמו למעלה רק עם anchor (עם משקלים).

search_body_title_one – אם אורך השאילתה שווה ל1 החיפוש התבצע ע"י הפונקציה search_title אם לא יתבצע search_body_title_with_weigh-

Search_BM25_for_some_index- פונקציית חיפוש 25BM בדומה לעבודה 4.

search_body_title_with_weight_BM25 –בדומה לפונקציה search_body_title_BM25_title, רק שהחיפוש מתבצע על פונקציית bm25 ולא החיפושים הראשונים (הtitle וה-text).

search_body_title_with_ _BM25_title – אם אורך השאילתה שווה ל1 – החיפוש יעשה ב search title אם לא, **ב**search_body_title_with_weight_BM25 .

Search_BM25 – מתבסס על bm25 בתוספת שינוי אקראי – הסברנו למעלה.

Search_BM25_with_condicion – הפונקציה שבסוף נבחרה, עם תנאי על המסמכים כך שהתדירות של המילה מהשאילתה מאותו מסמך תהיה גבוהה מ-50.