דו"ח פרויקט – איחזור מידע

מגישים: yulev@post.bgu.ac.il 325120384

katzmax@post.bgu.ac.il 322406604

github repo:

https://github.com/YuvalLevi1/IR_PROJ.git

Storage bucket:

https://console.cloud.google.com/storage/browser/proj_bucket1

List all index files with human-readable sizes(link to DRIVE):

https://drive.google.com/file/d/1aaoT_TKuqkt_Ge1rO0PzzrL78ljyJbGA/view?usp=share_link

תיאור הניסויים שהרצנו:

1). בהתחלה יצרנו גם index רגיל וגם index עם stemming ואחרי מספר בדיקות שונות הגענו למסקנה סופית שבלי stemming יוצא לנו טיפה יותר מהר ול -stemming היה לנו פחות דיוק לכן העדפנו לא להשתמש בstemming .

2). כאשר יצרנו את ה index לכל אחד מה- parts לא ידענו מה לשמור בנוסף לאינדקס הרגיל של אינדקס הפוך עד שלא ניסינו להריץ את פעולות החיפוש ולמצוא פתרונות להקטין את כל הפעולות שקורות בזמן ריצה (להקטין את זמן החזרת המסמכים מהשאליתא) ולשים אותם לאינדקס כך שבזמן offline ירוץ יותר זמן אך בזמן ריצה ירוץ בזמן קטן בהרבה. בין היתר הוספנו DL שעזר לנו מאוד וLEN CORFUS. לא שמרנו דברים מיותרים באינדקס שלא צריך בחישובים שלנו. הרצנו עם ובלי והיה הבדל גדול מאוד בזמני ריצה.

3). כשחישבנו את ה search body - היה כמה ניסויים שערכנו לגבי הנרמול של ה - tf וה – idf בהתחלה כשלא התשמשנו בtf and idf מנורמלים יצאו תוצאות פחות טובות אבל כשהשתמשנו בהם התוצאות ישתפרו.

4). בשביל פעולת ה -search היינו צריכים לכלול את המשקלים גם של ה -body וגם מה- title וגם מה-anchor אבל את ה -body חישבנו ודירגנו בעזרת cosine similarity לעומת ה -title וה -anchor שבהם עשינו דירוג בינארי (binary rank) ולכן ניסינו דרכים שונות לשקלל ציון. לדוגמא, ניסינו לתת משקלים על סמך הערך שיוצא משלושת הפונקציות האלה בין 0 ל 1 ולתת להם נרמול כך כדי שנוכל לסכום את כולם ביחד.
לאחר ניסויים רבים בנושא הגענו למסקנה כי לסכום את המשקלים על פי ציון לפי המיקום של התוצאה שחזרה בכל אחד מהparts כך שיהיה בסדר יורד מ100 שזה המשקל המקסימלי שאפשר לקבל על מאה מסמכים עד 0 ז"א שהמסמך אחרון ברשימה, עדיפה מכולם ומחזירה תוצאות משמעותיות יותר טובות מהשאר ולכן השתמשנו בו.

5). ניסינו להשתמש בחבילת WordNet שנלמדה בכיתה שמחזירה לכל מילה את המילים הנרדפות שלה (היא יכולה להחזיר כמה מילים נרדפות למילה מסוימת בשפה האנגלית). יש שם קרוב ל144000 מילים עם המילים נרדפות שלהם בכך ניסינו להרחיב את השאילתא ולמצוא עוד מסמכים רלוונטים שקשורים לנושא. אך גילינו כי החבילה הזו מחזירה recall גבוה ומקטינה את ה percision ולכן החלטנו לא להשתמש בחבילה זו כי אנו נמדדים על הpercision יותר.

6). עוד ניסויים שעשינו היו עם המדד של bm25 בפונקציית ה-search עבור שאילתות קצרות ועבור שאילתות ארוכות. אחרי ניסויים רבים, מצאנו כי המדד מוצא טוב יותר בשאילתות ארוכות ומעלה את הpercision בהם. אך בשאילתות קצרות, המדד טועה ומחזיר תוצאות נמוכות יותר מאשר מדד הcosine similarity. לכן החלטנו להשתמש במדד bm25 רק עבור שאילתות ארוכות ובכך הגדלנו את המשקל שהבאנו לו במודל שלנו.

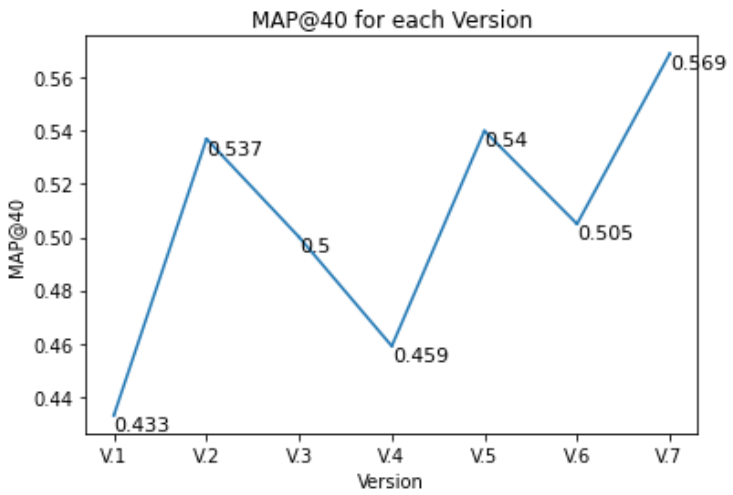
7). מצאנו כי כאשר מחשיבים את הanchor התוצאות קטנות של הpercision שאנחנו נבחנים עליהם להבדיל מtitle ו- body שבהם הייתה חשיבות גדולה עבור שאילתות קצרות וארוכות. אחרי בדיקות רבות, מצאנו כי עבור שאילתות קצרות צריך להביא משקל גדול לtitle כי כנראה שהמשתמש חיפש את הכותרת של מה שהוא חיפש ועבור שאילתות ארוכות שרוצות פירוט על המידע נביא משקל גדול יותר לbody .

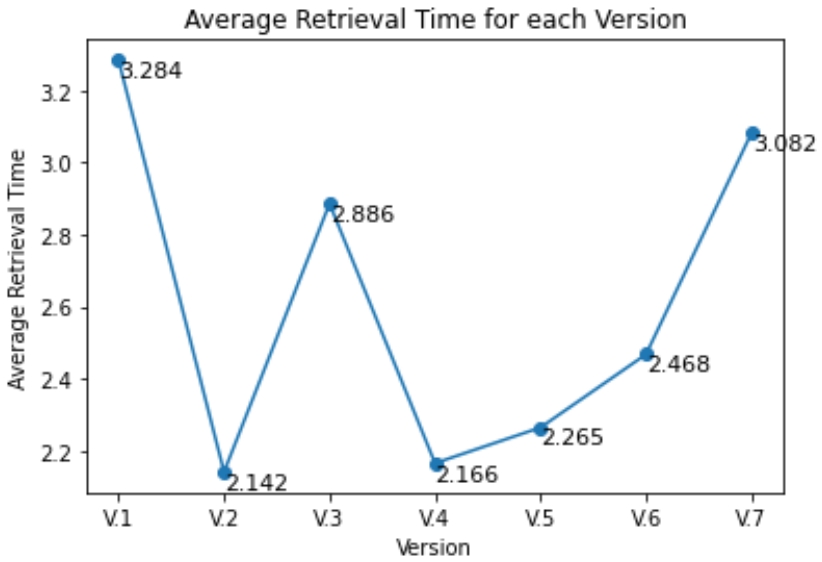
8). השיפור הבא שעשינו ל search -היה להוריד את כמות המילים שחוזרות מהשאילתא ובכך לפגוע ב recall אך להגדיל את הpercision. אחרי ניסויים בנושא, הגענו למסקנה ש80 מסמכים עבור שאילתות קצרות עדיף ועבור שאילתות ארוכות להחזיר 60 מסמכים עדיף. המסמכים שהציון שלהם הוא מתחת ל60 הללו בטוח לא רלוונטיים ולכן אם ניקח רק את המסמכים שהם מעל הסף יהיו לנו יותר מסמכים רלוונטיים ממסמכים לא רלוונטיים, דבר זה גם שיפר את הpercision שלנו.

9). השיפור האחרון שעשינו היה להוסיף עוד סידור לתוצאות מה -search אחרי הסידור הראשוני של המסמכים על פי כל הממדים. רק שהפעם בעזרת סידור של ה-pagerank ואז סידור בעזרת ה-pageviews. אחרי כמה בדיקות גילינו כי, להוסיף את הסידור שלהם העלה לנו הרבה תוצאות רלוונטיות שיופיעו במקומות הראשוניים והדבר שיפר את ה map משמעותית.

בדקנו את איכות המנוע לאורך כל הפרויקט דרך map@40 מדד זה נבחן בבדיקות ולכן הרגשנו שזה המדד הכי אובייקטיבי לנו ללכת על פיו.

<u>גרפים ביצועים עבור כל גרסה וזמנים ממוצעים לכל גרסה:</u>





<u>דוגמאות לתשובות:</u>

<u>דוגמא לשאילתא שיצא בה תוצאות טובות מאוד:</u>

(Air Jordan', 1.627821922302246, 1.0')

כמו שאפשר לראות השאילתה הכי טובה שלנו הייתה :
"Air Jordan":

כמו שניתן לראות ב 10 התוצאות הכי טובות כולם באמת קשורות לשאילתא דרך המילים שיוצאות ולא המשמעות של המילים. בנוסף, ניתן לראות כי התוצאה הראשונה בחיפוש יצאה Air Jordan כמו שהמשתמש חיפש לכן הוא לא ימשיך לחפש הלאה. מכאן ה percision מאוד גבוהה.


```
[
  1394509,
  "Air Jordan"
],
[
  58209447,
  "Air Jordan (airline)"
],
[
  59712869,
  "List of snakes of Jordan"
],
[
  27044653,
  "List of Doctor Slump episodes"
],
[
  56669626,
  "Air Italy S.p.A."
],
[
  28155315,
  "Air Arabia Jordan"
],
[
  59986325,
  "Tla' Al-Ali, Umm Al-Summaq and Khalda area"
],
[
  19660655,
  "Jordan International Air Cargo"
],
[
  59986327,
  "Shafa Badran area"
],
[
  59986394,
  "'Ayy"
],
```

דוגמא לשאילתא שיצא בה תוצאות גרועות:

```
('How do you make gold', 3.170180082321167, 0.0)
```

כמו שניתן לראות עבור השאילתה"How do you make gold" :

המנוע חיפוש שלנו לא עבד טוב בכלל. מה שלא עבד טוב בחיפוש של השאילתה הזו היה שהתשובות הן אינן קשורות לנושא זהב והם היו תשובות אחרות בנושאים לא קשורים. הוא התמקד במילה make ולא קישר אותה עם המילה gold ולכן אף תשובה אינה רלוונטית. בנוסף, הוא לא הבין את תוכן השאליתא.


```
[
  [
    26181063,
    "Make"
  ],
  [
    36750281,
    "Chrysus"
  ],
  [
    61700533,
    "Gold digging"
  ],
  [
    3421573,
    "E175"
  ],
  [
    40073388,
    "Sepon mine"
  ],
  [
    6904786,
    "United (Marian Gold album)"
  ],
  [
    2779990,
    "Harmony Gold"
  ],
  [
    12258139,
    "Claim jumping"
  ],
  [
    31070208,
    "Price of gold"
  ],
  [
    31057826,
    "TAPSO"
  ],
```


מסקנות:

המנוע חיפוש שלנו לא יודע למצוא את הנושא בשאלה ואת הקישור בין המילים ולכן עבור השאילתה הראשונה יצאו לנו תוצאות טובות כיוון שגם AIR וגם JORDAN נמצאים בעמודים של הויקיפדיה ולכן עבור השאילתה "Air Jordan" יצאו לנו תוצאות ממש טובות . ועבור השאילתה "How do you make gold" הנושא היה זהב אבל המנוע חיפוש ומצא סתם דברים שקשורים לזהב בלי להבין את התוכן של השאילתא. בנוסף, החזיר דברים עם הפועל לעשות שזה לא הנושא של השאילתא.

כדי לשפר את הבעיה הזו ניתן לאמן מודל שידע כמה מילים קשורות אחת לשנייה וככה הוא יוכל להביא תוצאות שיותר קשורות לנושא של השאלה ולא " ללכת לאיבוד".

ניסינו להשתמש באחד המודלים של למידת מכונה שנקרא Word2Vec אך הקריאה של הקבצים לBINS היתה מסובכת קצת ומפאת חוסר הזמן לא הספקנו לעשות את זה. בנוסף, חשבנו גם על Doc2Vec אשר יחזיר מסמכים דומים.