ניהול נתונים באינטרנט – פרויקט 2 – קובץ תיעוד

תום רוטנברג 313196727

עופר טלוסטי 311396303

חלק א' – בניית אינדקס הפוך

עבור פקודת "$create\ index$" יחד עם ניתוב לתיקיית קבצים אנו מפעילים את הפונקציה $createIndex()$ אשר תקרא את קבצי ה-xml בתיקייה, תחלץ מתוכם את המסמכים ותעדכן את המילון אשר ישמש אותנו למענה על השאלות בחלק הבא של הפרויקט.

נתחיל מתיאור המילון אשר ישמש אותנו להמשך המימוש:

1. **מילון המסמכים ($docs\_dict$)** – מילון זה ישמש אותנו למצוא את כלל הפרטים הנחוצים לנו בהינתן מזהה מסמך ($RECONRD\_NUM$). המילון בנוי באופן ההיררכי הבא:
המפתח של המילון הראשי הינו $RECORD\_NUM$ והערך שלו הינו מילון נוסף הכולל מספר שדות:
   - אורך הוקטור ($VECTOR\ LENGTH$) – עבור חישוב $cosin\ similarity$ אנו נדרשים לנרמל את התוצאה בגודל הוקטור של המסמך ושל השאילתא. על מנת לחסוך חישובים בזמן ריצה (והמתנה של המשתמש) אנו שומרים את גודל הוקטור של המסמכים השונים במילון כך שנוכל לשלוף אותו מבלי לבצע חישובים נוספים.
   - תדירות מרבית ($MAX\ FREQ$) – עבור חישוב פרמטר ה-$TF\ (Term\ Frequency)$ נרצה לנרמל את תדירות המילים במסמך לפי תדירות המילה השכיחה ביותר במסמך. לכן נרצה לחשב ולשמור ערך זה במילון ולהשתמש בו בעת חישוב הפרמטר TF.
   - ספירת מילים במסמך ($WORD\ COUNT\ IN\ DOC$) – עבור חישובי $BM25$ נרצה לשמור את מספר המילים הכולל בכל מסמך.
   - מילון מילים במסמך ($WORDS\ IN\ DOX$) – במילון זה המפתח יהיה מילה המוכלת במסמך הנתון והערך המתקבל יהיה הפרמטר $TF - IDF$ של המילה במסמך הספציפי.

להלן דוגמה מהמילון להמחשת המבנה שלו:

```
"docs_dict": {
    "1": {
        "vector_length": 0.5029287715746582,
        "max_freq": 6,
        "word_count_in_doc": 119,
        "words_in_doc": {
            "pseudomona": {
                "tf-idf": 3.279258724379981
            },
```

2. **מילון המילים ($words\_dict$)** – מילון זה ישמש אותנו למצוא את כלל הפרטים הנחוצים לנו בהינתן מילה במאגר המילים הכולל של "מנוע החיפוש שלנו" והוא בנוי באופן ההיררכי הבא:
   - תדירות מסמכים הפוכה (IDF) – נרצה לחשב פרמטר זה עבור כל מילה כחלק מחישוב המשקלים של המילים במסמכים השונים ($TF - IDF$).
   - מילון המסמכים המכילים את המילה ($DOC\ CONTAIN\ WORD$) – במילון זה המפתח הינו מזהה המסמך ($RECORD\_NUM$) ולו שני ערכים:
     - ספירת המילה במסמך ($count\ word\ in\ dox$) – פרמטר זה ישמש לטובת חישוב הפרמטר TF.

- <u>תדירות המילה במסמך (TF)</u> – נרצה לחשב פרמטר זה עבור כל מילה במסמך כחלק מחישוב המשקלים של המילים במסמכים השונים ($TF - IDF$).

להלן דוגמה מהמילון להמחשת המבנה שלו:


```
"words_dict": {
    "pseudomona": {
        "idf": 3.935110469255977,
        "doc_contain_word": {
            "1": {
                "count_word_in_doc": 5,
                "tf": 0.8333333333333334
            },
```


עתה נחזור לאופן פעולת הפונקציה $createIndex()$:

- בעבור כל קובץ xml בתיקיית האינפוט נקרא לפונקציה $parseFile$ אשר תמצא את כלל המסמכים (ישות $RECORD$).
- עבור כל מסמך $RECORD$ נבצע את הפעולות הבאות:
  - נחלץ את מזהה המסמך ($RECORD\_NUM$) ונאתחל רשומה מאופסת במילון המסמכים ($docs\_dict$).
  - נחלץ את תוכן כל המסמכים באמצעות הפונקציה $extractWords$ אשר מקבלת מזהה מסמך ומחזירה מחרוזת הכולל את כלל המילים במסמך תחת הישויות הבאות: $Title, Abstract, Extract, Major\ and\ minor\ subjects$ אשר נמצאו רלוונטיות לצרכי משימה זו. כמו שנלמד בשיעור אחד הפרמטרים לשיפור הביצועים במנגנון החיפוש של גוגל החלטנו למשקל את האפקט של מילים המופיעות בכותרת ובנושאי המסמך לעומת המילים במסמך עצמו. לאחר מספר ניסיונות הוחלט על אופן המשקול הבא:
    - כותרת ($Title$) – משקל 10.
    - נושאי המסמך ($Major\ Subject, Minor\ Subject$) משקל 5.
    - המילים במסמך עצמו ($Abstract, extract$) – משקל 1.

    נציין כי בשלב זה אנו מבצעים ניקוי ראשוני על המילים במסמך באמצעות הפונקציה $strip()$.
  - נתאחל פרמטר לוקאלי max_freq אשר ישמור את תדירות הגבוהה ביותר שזיהינו עד כה למילה במסמך והוא יעודכן בכל איטרציה של הלולאה הבאה.
  - נעבור בלולאה על כלל המילים אשר חולצו מהמסמך ונבצע את הפעולות הבאות:
    - ננקה את המילה באמצעות הפונקציה $parseWord()$ – פונקציה זו תנקה את המילים מסימני פיסוק, תבצע פעולת $stemming$ ותקטין את כלל האותיות במילה לאותיות קטנות. כמו כן, נבצע בדיקה האם המילה כלולה במאגר $stopWords$ ולכן נרצה שלא להכניסה למאגר שלנו (כפי שנלמד בשיעור).
    - נציין כי פעולות הניקוי נעשות תוך שימוש בספריה $ntlk$ בדגש על מאגר ה-$stopWords$ והחבילה $PorterStemmer$.
    - במידה והמילה ואלידית (לא כלולה במאגר ה-$StopWords$) נרצה להזין את המילונים שלנו באמצעות שימוש בשתי הפונקציות הבאות:
      - $insertToDocsDict()$ – הפונקציה תאתחל רשומה עם המסמך והמילה הנתונים עם ערך $TF - IDF$ מאופס.
      - $insertToWordsDict()$ – הפונקציה תבדוק האם קיימת רשומה עם המילה הנתונה, במידה ולא תזין רשומה מאופסת. לאחר מכן, תבדוק המסמך

נמצא במילון הפנימי ($Dox\_contain\_word$), במידה ולא תזין רשומה מאופסת גם כן. לבסוף, הפונקציה תעלה את המונה $Count\_Word\_In\_Doc$ במילון.

- נעדכן את מונה המילים במסמך ($Word\_Count\_In\_Doc$) ונחשב את שכיחות המילה במסמך. במידת הצורך נעדכן את התדירות המקסימלית הלוקאלית שאתחלנו לפני הכניסה ללולאה.

- בסיום המעבר על כלל המילים במסמך, נזין למילון את תדירות המילה השכיחה ביותר (MAX_FREQ) ונקרא לפונקציה $calcTFValues()$.
- הפונקציה $calcTFValues()$ עוברת על כלל המילים במסמך הנתון, מנרמלת את תדירות המילים במסמך לפי התדירות המקסימלית שחושבה ושומרת את הערך המתקבל במילון $words\_dict$ (כפי שהוסבר מעלה).

בסיום המעבר על כלל הקבצים ופרסורם, המילונים מכילים בתוכם את כלל המסמכים והמילים אך ישנם מספר פרמטרים אשר נצטרך וטרם חושבו – $IDF, TF-IDF, VECTOR\_LENGTH$:

1. הפונקציה $cacl\_IDF\_And\_TFIDF\_Values$ אחראית לחישוב זוג הפרמטרים הראשונים והזמנתם למילונים כפי שהוצגו מעלה. נציין כי אופן החישוב של הפרמטרים הוא כפי שנלמד בכיתה ולא נכנסנו במסמך זה לאופן החישוב שלהם).
2. הפונקציה $calcSqrtVectorLength()$ אחראית על חישוב אורך הוקטור עבור כלל המסמכים במאגר שלנו על מנת לחסוך בחישובים "בזמן אמת".

לבסוף נשמור את המילונים לקובץ $json$ בשם "$vsm\_inverted\_index.json$" כנדרש.

<u>חלק ב' – בניית מנגנון האחזור</u>

מנגנון האחזור מופעל על ידי העברת הארגומנט "$query$", אחריו מציינים את השיטה לפיה מעוניינים שהמערכת תדרג את התוצאות ($tf-idf$ / $BM25$), ניתוב לקובץ ה- $inverted\ index$ אותו בנינו בחלק א' ולבסוף את השאילתא עצמה.

תחילה אנחנו מעבדים את השאילתה ע"י פרסור כל מילה בה באותה הדרך בה פרסרנו את הקורפוס שלנו (פונקציית $parseWord()$ אשר תוארה לעיל). לאחר מכן אנו יוצרים את וקטור השאילתא על ידי נירמול מספר המופעים של כל מילה מפורסרת והכפלה של ערך זה בערך IDF אותו אנחנו לוקחים מהמילון $words\_dict$ שהוא חלק מה- $Inverted\ index$. נציין כי גם את וקטור השאילתא אנו מתחזקים באמצעות מבנה של מילון משיקולי יעילות.

אנו מתעלמים ממילים/מונחים בשאילתא אשר אינם מופיעים כלל ב- $words\_dict$. הסיבה לכך היא שמאחר ואף מסמך בקורפוס לא מכיל את המונח אז לכל מסמך בקורפוס ה- $term\ frequency$ של המונח יהיה אפס ולא תיהיה לו תרומה לציון הסופי.

כעת יש בידנו את וקטור המשקולות של השאילתא, ללא המונחים אשר המשקולת שלהם היא אפס, עיקרון זה מיעל מאד את החישוב שכן בעולם תוכן של אחזור מידע הוקטורים נוטים להיות מאד $sparse$.

כעת לכל מונח בוקטור של השאילתא אנחנו מחלצים את מזהי המסמכים מהקורפוס אשר מכילים מונח זה (באמצעות ה- $word\_dict[tem][DOC\_CONTAIN\_WORD]$), לכל מסמך כזה אנחנו כופלים את

המשקולות של וקטור השאילתא ושל המסמך עבור המונח הנ"ל ומוסיפים את זה לציון המשוכלל עבור אותו מסמך.

בסוף הריצה אנחנו מנרמלים את הציון הסופי של כל מסמך באורך וקטור שלו על מנת לנטרל הטייה לטובת מסמכים ארוכים בהינתן שאילתא, וגם אנחנו מנרמלים באורך וקטור השאילתא וזאת על מנת שנוכל להשתמש בפרמטר של ערך סף מינמלי של ציון על פני שאילתות שונות.

בשאילתות המשתמשות בשיטת $BM25$ לדירוג התוצאות תהליך החישוב דומה מאד אך במקרה זה לא מדובר בכפל של משקולות בין וקטור השאילתא לוקטור של המסמך, אלא בסכימה של מכפלות עבור כל מונח בשאילתא.

נציין כי במקרה זה את המכפלה הנ"ל ניתן לחשב בקלות ובמהירות בהינתן שנתונים לנו כבר ערכי ה- $term\ frequency$ ומספר המסמכים בהם כל מונח מופיע ולכן החישוב הנ"ל מתבצע $Online$ בהינתן שאילתא. במידה והיינו מבצעים את החישוב $offline$ מבעוד מועד הדבר היה חוסך מעט מאד זמן בריצת השאילתה אבל מגדיל את זמן בניית ה- $Inverted\ Index$ כי היינו צריכים לחשב את הערכים האלה עבור כל מסמך וייתכן שלא כל מסמך יתושאל בשיטה זו, והסיבה השניה שקשורה בעיקר באופטימיזציה היא שהערכים הנ"ל מושפעים מקביעת ערכי הפרמטרים k_1, b ובאופן של חישוב $Online$ לא צריך לבנות את ה- $Inverted\ Index$ מחדש בכל שינוי של אחד הפרמטרים.