# דו"ח מנוע חיפוש – חלק ב'

**הוגש ע"י –**

**מיכל טלמור 312461080**

**דורון שמאי - 312538325**

# חלק ראשון

## א. הסבר מפורט על אופן פעולת המנוע:

כאשר מריצים את התוכנית נפתח ממשק המשתמש ובו המשתמש יכול לעשות מספר דברים:
אופציה ראשונה: להכניס 2 נתיבים, אחד של מאגר הקבצים ואחד של המיקום שבו ישמור קבצים הposting.
אופציה שנייה: להכניס רק נתיב אחד, שהוא המיקום שבו קיימים קבצי הposting והמילונים.
באופציה הראשונה נלחץ על כפתור הstart ואז המנוע ממש ירוץ על כל הcorpus וייצור את קבצי הposting והמילונים ואז יהיה ניתן להמשיך. באופציה השנייה (במידה וקבצי הposting והמילונים כבר קיימים) ניתן ללחוץ על כפתור load dictionary ואז התוכנית תטען לזיכרון את כל המידע הנחוץ להמשך ריצת התוכנית.

לאחר שטענו את כל המילונים לזיכרון והשגנו את כל המידע הנחוץ לנו לשם אחזור שאילתות, יש לנו מספר דרכים להריץ שאילתות במנוע:
דרך ראשונה: ניתן לכתוב שאילתה בתיבת הטקסט "search" ואז ללחוץ על כפתור ה run. דבר זה יריץ את השאילתה שכתבנו ובעזרת המחלקות searcher ו- ranker ננתח את השאילתה בדומה לניתוח שביצענו על המסמכים ב corpus ונחזיר את 50 המסמכים הרלוונטיים ביותר לשאילתה שכתבנו. המסמכים יחזרו בצורה מדורגת (תפקידה של מחלקת ranker).
מחלקת הranker תדרג את התשובות של השאילתות, כלומר היא תדרג את המסמכים הכי רלוונטיים לשאילתה הספציפית.
הדרך השנייה: ניתן לטעון קובץ של שאילתות ובעזרת כפתור ה browse במיקום "choose queries file" , ואז המנוע ירוץ על כל השאילתות אשר נמצאות בקובץ ויבצע עליהן את אותן הפעולות כמו שהיה מבצע על שאילתה בודדת שנכתבה בשורת ה run.
בסוף ההרצה של שאילתה בודדת דרך שורת ה run או הרצה של מספר שאילתות מתוך קובץ יופיע על המסך חלון ובו רשימה של 50 המסמכים הכי רלוונטיים לשאילתות שהרצנו. נוכל לבחור כל מסמך שנרצה ועבורו ללחוץ על כפתור "identify entities" ופעולה זו תחזיר לנו חלון עם 5 הישויות הדומיננטיות ביותר במסמך.
במחלקה Runker נחזיק משתנה מסוג Dictionary. המחלקה Dictionary היא המחלקה שאליה נטענים המילונים לאחר תהליך ה load dictionary או לאחר תהליך האינדוקס.
מחלקת ה Ranker שואבת ממחלקת ה Dictionary את המילונים לתוך שני HashMap.
HashMap אחד עבור מילון המילים וHashMap שני עבור מילון המסמכים.
המחלקה Searcher מחזיקה אובייקט של המחלקה Ranker ובכך המחלקה Searcher יכולה לדרג את המסמכים שהחלקה Searcher מחזירה עבור שאילתה ספציפית.

המחלקות שהוספנו –

- Searcher – מחלקה זו תפקידה לקבל שאילתה מהמשתמש במגוון דרכים, באמצעות טקסט חופשי ובאמצעות נתיב לקובץ המאגד מספר שאילתות שונות, ולהחזיר את המסמכים הרלוונטים מדורגים לפי הסדר ולכתוב קובץ טקסט המכיל את התוצאות בדיסק בנתיב שהתקבל מהמשתמש.

- Ranker – מחלקה זו תפקידה לקבל רשימה של terms המרכיבים את השאילתה הרלוונטית ולבצע דירוג למסמכים המכילים את הterms הרלוונטים לפי אלגוריתם שמימשנו באמצעות קבצי הposting והמילון הנמצאים בדיסק ומכילים את הנתונים.

## ב. פירוט המחלקות הרלוונטיות בחלק זה:

- Searcher –

מחלקה זו היא המחלקה בה אנחנו מבצעות את הקבלה של השאילתה מהמשתמש. בתוך המחלקה ישנו מופע של מחלקה נוספת "Ranker" (יפורט בהמשך), שמבצע את הדירוג לפי השאילתות המתקבלות.
בנאי המחלקה מקבל שני פרמטרים – מופע של Dictionary המחזיק את המילונים (מילון עבור המסמכים ומילון עבור ה-terms) ונתיב לקבצי ה-posting שלפיהם נבצע את דירוג המסמכים עבור השאילתה.
שיטות המחלקה הן –

1. **StringQuery –** שיטה זו מקבלת שאילתה מהמשתמש המוצגת בצורת "טקסט חופשי" בחלון חיפוש, בשיטה זו אנו מחלקים את השאילתה ל-terms ומבצעים על כל אחד מהם "פירסור" באמצעות הפונקציה "parseQuery", לאחר ביצוע הפירסור הterms נשלחים למחלקה "Ranker" ומחזירה ArrayList של כל המסמכים הרלוונטים מדורגים לפי הסדר. בשיטה זו גם נבדוק האם יש צורך לבצע טיפול סמנטי בterms שהתקבלו, ובמידה וכן נפעיל את השיטה "semnticValues" שתחזיר לנו terms נוספים לביצוע דירוג המסמכים על פיהם. אנו נותנים בצורה רנדומלית מספר לשאילתה שלא יתנגש עם מספרי שאילתות קודמות ושולחות את פרטי השאילתה ואת המסמכים המדורגים לכתיבה לקובץ טקסט בזיכרון באמצעות הפונקציה "writeResults" שתפורט בהמשך.

2. **PathQuery –** שיטה זו מקבלת נתיב מהמשתמש לקובץ טקסט המכיל רשימה של שאילתות לפי פורמט מסויים. שיטה זו מחלצת את השאילתות המופיעות בקובץ טקסט אחת אחרי השניה, היא שומרת עבור כל שאילתה את המספר שלה, ואת הterms של השאילתה עצמה. לאחר פיצול הterms הם נשלחים לשיטה "stringQueryFromPath" שמחזירה רשימה של מסמכים מדורגים ותפורט בהמשך. בנוסף שיטה זו שולחות את פרטי השאילתה ואת המסמכים המדורגים לכתיבה לקובץ טקסט בזיכרון באמצעות הפונקציה "writeResults" שתפורט בהמשך.

3. **stringQueryFromPath-** שיטה זו מקבלת את הterms של השאילתה לאחר שחולצה מהנתיב, בשיטה זו אנו מחלקים את השאילתה ל-terms ומבצעים על כל אחד מהם "פירסור" באמצעות הפונקציה "parseQuery", לאחר ביצוע הפירסור הterms נשלחים למחלקה "Ranker" ומחזירה ArrayList של כל המסמכים הרלוונטים מדורגים לפי הסדר. בשיטה זו גם נבדוק האם יש צורך לבצע טיפול סמנטי בterms שהתקבלו, ובמידה וכן נפעיל את השיטה "semnticValues" שתחזיר לנו terms נוספים לביצוע דירוג המסמכים על פיהם.

4. **parseQuery –** שיטה זו מקבלת מערך המכיל את הterms הרלוונטים עליהם נרצה לבצע פירסור. השיטה עוברת על כל אחד מהם, ובאמצעות אובייקט של המחלקה "parser" מחזירה את הערך המפורסר של כל term שהתקבל.

5. **writeResults –** שיטה זו מקבלת רשימה את הרשימה של המסמכים המדורגים ואת מספר השאילתה וכותבת את התוצאות לתוך קובץ טקסט לפי הפורמט המבוקש.

6. **semanticValues –** שיטה זו מקבלת מערך של כל הtemrs עליהם אנו רוצים לבצע שיפור סמנטי ומחזירה מערך חדש המכיל מילים נוספות הדומות מבחינה סמנטית למילים הקיימות. עבור כל trem, באמצעות המחלקה "Medallia", ובאמצעות נתיב מקומי המכיל את המילים הקשורות למרבית המילים במילון, ביצענו שליפה של 2 מילים נוספות הדומות מבחינה סמנטית לterm. את המילים הללו הוספנו לרשימת ה-terms המקוריים בשאילתה.

- Ranker –

מחלקה זו היא המחלקה בה אנו מבצעים את דירוג המסמכים לפי השאילתה הרלוונטית. מחלקה זו משתמשת באלגוריתם שפיתחנו ועפ"י פרמטרים שונים שיפורטו בהמשך, נדרג כל מסמך לפי מידת הרלוונטיות שלו למילים בשאילתה. מחלקה זו מקבלת בבנאי שלה את המילונים הרלוונטים (מילון המסמכים ומילון המילים) ואת הנתיב לקבצי הpostingשלפיהם נבצע את אלגוריתם הדירוג.

1. **rankDocsByQuery-** שיטה זו מקבלת את מערך של terms המרכיבים את השאילתה על פיהם נערוך את דירוג המסמכים. עבור כל term נגש לערך שלו בקובץ ה-posting בו הוא מופיע ונבדוק באיזה מסמכים הוא מופיע, עבור כל מסמך שבו הוא מופיע, נחלץ את הפרטים הרלוונטים דרך מילון המסמכים בשביל ונפעיל על המסמך את השיטה "rankBM25" הפועלת לפי אלגוריתם שיפורט בהמשך ומחזירה את ציון המסמך. נחזיק מבנה נתונים המאגד את הציון של כל מסמך וכל פעם שיש term שמוסיף ציון למסמך שמופיע נעדכן את הערך לערך הישן במבנה הנתונים. עבור מסמכים שלא מופיעים ניצור ערך במבנה הנתונים ונכניס את הציון שהוחזר. לבסוף, נחלץ ממבנה הנתונים את 50 המסמכים שקיבלו את הציון הגבוה ביותר ונחזיר אותם.

2. **rankBM25-** שיטה זו מקבלת את פרטי המסמך עליו אנו רוצים להחזיר ציון למידת הרלוונטיות שלו עבור term המופיע בשאילתה, ואת מספר המסמכים בהם מופיע הterm שעבורו נחזיר את הציון. השיטה מבצעת את חישוב הציון לפי האלגוריתם שיפורט בהמשך, היא מחלצת את כל הפרטים הרלוונטים לאלגוריתם מהארגומנט שהתקבל באמצעות מילון המסמכים שהתקבל בבנאי ומפרמטרים שהוגדרו במחלקה עצמה.

## ג. האלגוריתמים הכלולים במנוע:

### 1. אלגוריתם הדירוג –

האלגוריתם בו השתמשנו על מנת לדרג את המסמכים הוא האלגוריתם "BM25".
אלגוריתם זה נותן ציון עבור כל צמד של מילה בשאילתה ומסמך שבו המילה מופיעה.
במידה ומסמך קיבל ציון עבור יותר ממילה אחת בשאילתה, הציון הסופי שלו יהיה סכום כל הציונים שקיבל עבור כל המילים שהופיעו בו. אופן קבלת הציון דרך האלגוריתם הוא בצורה הבאה –

$$\text{score}(D,Q) = \sum_{i=1}^{n} \text{IDF}(q_i) \cdot \frac{f(q_i, D) \cdot (k_1 + 1)}{f(q_i, D) + k_1 \cdot \left(1 - b + b \cdot \frac{|D|}{\text{avgdl}}\right)},$$

נפרט על כל אחד מהפרמטרים –

$$\text{IDF}(q_i) = \log \frac{N - n(q_i) + 0.5}{n(q_i) + 0.5},$$

- N – כמות המסמכים במאגר המסמכים.
- D – אורך מסמך.
- Avgdl – אורך ממוצע של מסמך במאגר.
- $f(qi, D)$ – ערך מנורמל של TF-IDF, מספר הפעמים שהמילה qi מופיעה במסמך D חלקי אורך ממוצע של מסמך במאגר. (הנרמול נעשה על מנת שמסמך קצר יקבל את אותה הערכה כמו מסמך ארוך באופן יחסי לפי מספר המופעים של הterm במסמך).
- $n(qi)$ – כמות המסמכים שהמילה מופיעה בהם במאגר.
- $k1$ - ערך קבוע שבחרנו באמצעות הרצות שונות וחיפוש אחר ערך שיחזיר את כמות התוצאות הרלוונטיות הגדולה ביותר.
- $b$ - ערך קבוע שבחרנו באמצעות הרצות שונות וחיפוש אחר ערך שיחזיר את כמות התוצאות הרלוונטיות הגדולה ביותר.

### 2. אלגוריתם למציאת 5 היישויות הדומיננטיות במסמך –

על מנת לחשב את 5 היישויות הדומיננטיות במסמך, נתנו ציון לכל אחת מהישויות בדרך הבאה – עבור כל יישות חישבנו את כמות הפעמים שהיא מופיעה במסמך. בנוסף מצאנו את ערך ה- maxTF עבור הישויות במסמך, כלומר את מספר הפעמים שמופיעה הישות שמופיעה הכי הרבה פעמים במסמך. לכל ישות, נרמלנו את מספר הפעמים שהיא מופיעה במסמך בערך ה-maxTF שחישבנו. הציון שקיבלה כל יישות הוא הערך המחושב ע"י נרמול זה. ה5 יישויות שיחזרו מהאלגוריתם יהיו 5 היישויות שיקבלו את הציון הגבוה ביותר.

דוגמאות:

1. עבור המסמך FBIS3-3119 –
הישויות שיוחזרו לפי הסדר הן:

1.0 United States
0.26666666666666666 South Africa
0.26666666666666666 Hong Kong
0.2 State Christophers
0.2 Qian China

2. עבור המסמך FT941-7901 –
הישויות שיוחזרו לפי הסדר הן:

1.0 Asia Watch
0.5 A Chinese
0.5 Favoured Nation
0.5 Most Favoured Nation
0.5 Amnesty International

נראה כי הדירוג שקיבלה כל יישות היא באופן יחסי לאחרות, כך שהיישות שמופיעה הכי הרבה פעמים מקבלת דירוג 1, ושאר הישויות מנורמלות לפי הכמות שהיא מופיעה.

### 3. אלגוריתם לשיפור סמנטי –

מהתוצאות שהוחזרו מהדמיון הסמנטי שהופיע בקובץ הטקסט המצורף המכיל את רשימה של מילים ומילים בעלות אותה משמעות סמנטית לקחנו רק 2 ערכים נוספים עבור כל term שהופיע בשאילתה. הסיבה לכך היא שחלק מהמילים לאו דווקא מתאימות להקשר הספציפי של שאר המילים בשאילתה ועל מנת לא לפגוע בתוצאות ובזמן שלוקח לשליפה (שכן כל מילה נוספת מצריכה מעבר על קובץ הפוסטינג הרלוונטי ושליפת הערכים), העדפנו להגביל את כמות המילים הנוספות ל-2 בלבד. בנוסף למילים הנוספות נתנו ערך נמוך יותר בחישוב הסופי של האלגוריתם כדי שתהיה להן פחות השפעה מאשר למילים המקוריות שמופיעות בשאילתה.

## ד. הסבר על הנתונים במילון ובקבצי ה-posting:

### 1. המילון הראשי של ה-terms –

דוגמא לכמה ערכים מהמילון:

```
alloys 169 239CRLF
Alloys Foundry 1 1CRLF
Alloys MSampA 2 2CRLF
ALLPASS 3 6CRLF
ALLPORT 2 2CRLF
```

- הערך הראשון – מציין את מספר המסמכים בהם הterm מופיע במאגר המסמכים.
- הערך השני – מציין את מספר המופעים הכללי של ה-term במאגר.

נצטרך את שני הערכים הנ"ל לחישוב האלגוריתמים לדירוג המסמכים.
הערך הראשון רלוונטי לחישוב שדה ה- $n(qi)$ באלגוריתם BM25.
הערך השני רלוונטי לנרמול כמות המופעים של term כללי במאגר לפי מספר המופעים של term הנפוץ ביותר במאגר. למשל נעשה שימוש בנרמול מהסוג הזה באלגוריתם למציאת 5 ישויות דומיננטיות.

## 2. המילון של המסמכים –

דוגמא לכמה ערכים מהמילון:

```
FT941-7911 8 150 192 ,1.0 L450bn Pounds,1.0 The Italian,1.0 L1,800bn Pounds
FT941-7910 7 176 220 ,1.0 Gasprom Russias,1.0 Financial Times,1.0 Mr Chernomyrdin,1.0 Russia An,1.0 European Bank
FT941-7913 13 194 258 ,1.0 The EU,1.0 Hungary By,1.0 Ireland Norways,1.0 Spains European,1.0 Alps Because
FT941-7912 5 79 103 ,1.0 The French,0.5 Italy Spain
FT941-7915 5 194 238 ,1.0 Rose UN,1.0 New York,1.0 Hercegovina We,1.0 European Union Canada,1.0 Yesterday Serb
FT941-7914 10 171 244 ,1.0 IG Metall,0.8 Lower Saxony,0.2 Klaus Zwickel,0.2 Mr Klaus Zwickel
FT941-7917 5 184 239 ,1.0 The UK,0.5 Mr Alan Greenspan,0.5 Alan Greenspan,0.5 Reserve Board,0.5 Statistical Office
FT941-7916 6 156 188 ,1.0 London Dollars,1.0 Apr Dollars,0.5 Pounds Index,0.5 FT-A World Index,0.5 Treas Bills Yld
```

- הערך הראשון – מציין את מספר הפעמים שמופיעה המילה בעלת התדירות הגבוהה ביותר במסמך.
- הערך השני – מציין מספר המילים הייחודיות במסמך.
- הערך השלישי – מציין את אורך המסמך.
- הערך הרביעי – מציין את 5 הישויות הנפוצות ביותר במסמך ואת הציון שניתן להם בצורה מנורמלת כמו שפורט קודם לכן.

נצטרך את כל הערכים הנ"ל לחישוב האלגוריתמים לדירוג המסמכים.
הערך הראשון רלוונטי לנרמול כל מילה אחרת הנמצאת במסמך.
הערך השני רלוונטי לחישוב שדה באלגוריתם הדירוג כאשר נרצה לנרמל את כמות המילים בשאילתה מסויימת לעומת כמות המילים הייחודיות הקיימות במסמך באופן כללי.
הערך השלישי רלוונטי לחישוב שדה ה- D באלגוריתם BM25 לדירוג המסמכים.

## 3. קובץ Posting–

דוגמא לכמה ערכים מקובץ ה-Posting:

```
FAEROES~&FBIS3-60549 4 11 38 54 67 &FBIS4-19259 15 18 75 147 174 202 331 352 403 439 720 735 847 912 983 1560 &FBIS4-64930 1 86
FAEROESE~&FBIS4-19259 5 92 129 270 1035 1093 &FBIS4-42602 2 24 119 &FBIS3-18855 5 28 73 87 117 132&FBIS3-18854 4 222 249 271 304
FAEROESE-BRITISH~&FBIS3-18854 1 14
```

נשים לב שיש כאן רשימה של כל המסמכים בהם המילה מופיעה (נתונה רשימה חלקית כמובן לצורך הדגמה). עבור כל מסמך שהמילה מופיעה בו נשמור:

- הערך הראשון – כמות הפעמים שמילה מופיעה המסמך.
- הערכים הבאים – רשימת האינדקסים שהמילה מופיעה בהם במסמך.

נצטרך את כל הערכים הנ"ל לחישוב האלגוריתמים לדירוג המסמכים.
את כמות הפעמים שהמילה מופיעה במסמך נצטרך לחישוב שדה ה- $f(qi, D)$ באלגוריתם BM25.
את רשימת האינדקסים שהמילה מופיעה בהם במסמך נצטרך על מנת לתת דירוג שונה לכל מסמך בו המילה מופיעה, למשל – במידה והמילה מופיעה בחלק העליון של המסמך היא תקבל דירוג שונה ממילה המופיעה בחלקו התחתון של המסמך. בנוסף מסמך שבו מילים מהשאילתה שמופיעות קרוב אחת לשניה יקבל דירוג גבוה יותר ממסמך אחר.

## ה. שימוש בקוד פתוח:

במהלך העבודה השתמשנו בקוד פתוח על מנת לממש את השיפור הסמנטי.
השתמשנו ב-jar של המחלקה word2VecModel שלקחנו מהכתובת –
https://github.com/medallia/Word2VecJava
השימוש היה במחלקה Searcher בשיטה – semanticValue שתפקידה היה להחזיר רשימה של מילים בעלות אותה משמעות סמנטית למילים שהיא קיבלה.
באמצעות המחלקה שייבאנו יכולנו ליצור מופע שלה בקוד ולהשתמש בשיטות שלה ובכך ליצור רשימה של מילים רלוונטיות.

# חלק שני – הערכה של המנוע

**No Stemming**

| Query number | Query words | Query Precision | Query Recall | precision @5 | precision @15 | precision @30 | precision @50 | Query running time |
|---|---|---|---|---|---|---|---|---|
| 351 | Falkland petroleum exploration | 16/50 | 16/48 | 0.2 | 0.4 | 0.3 | 16/50 | 1.016 |
| 352 | British Chunnel impact | 11/50 | 11/246 | 0.4 | 0.4 | 0.3 | 11/50 | 1.01 |
| 358 | blood-alcohol fatalities | 13/50 | 13/51 | 0 | 0.3333 | 0.3 | 13/50 | 0.437 |
| 359 | mutual fund predictors | 0 | 0 | 0 | 0 | 0 | 0 | 0.983 |
| 362 | human smuggling | 4/50 | 4/39 | 0.2 | 0.2 | 0.1 | 4/50 | 0.654 |
| 367 | piracy | 10/50 | 10/185 | 0 | 0 | 0.0333 | 10/50 | 0.414 |
| 373 | encryption equipment export | 4/50 | 4/16 | 0 | 0.0667 | 0.1 | 4/50 | 0.665 |
| 374 | Nobel prize winners | 11/50 | 11/204 | 0.6 | 0.2667 | 0.3333 | 11/50 | 0.671 |
| 377 | cigar smoking | 10/50 | 10/36 | 0 | 0.0667 | 0.2 | 10/50 | 0.933 |
| 380 | obesity medical treatment | 4/50 | 4/7 | 0 | 0.2 | 0.1 | 4/50 | 0.672 |
| 384 | space station moon | 6/50 | 6/51 | 0.2 | 0.1333 | 0.1333 | 6/50 | 1.261 |
| 385 | hybrid fuel cars | 16/50 | 16/85 | 0 | 0.3333 | 0.3667 | 16/50 | 0.818 |
| 387 | radioactive waste | 0/50 | 0/73 | 0 | 0 | 0 | 0/50 | 0.413 |
| 388 | organic soil enhancement | 6/50 | 6/50 | 0 | 0.2 | 0.1333 | 6/50 | 0.767 |
| 390 | orphan drugs | 10/50 | 10/122 | 0.4 | 0.2 | 0.2333 | 10/50 | 0.352 |
| 15 | | 121/750 | 121/1241 | 0.1333 | 0.1867 | 0.1756 | 121/750 | 11.066 |

MAP (No Stemming) = 0.1074

**Stemming**

| Query number | Query words | Query Precision | Query Recall | precision @5 | precision @15 | precision @30 | precision @50 | Query running time |
|---|---|---|---|---|---|---|---|---|
| 351 | Falkland petroleum exploration | 10/50 | 10/48 | 0.2 | 0.3333 | 0.2333 | 10/50 | 3.425 |
| 352 | British Chunnel impact | 11/50 | 11/246 | 0.4 | 0.3333 | 0.3 | 11/50 | 1.014 |
| 358 | blood-alcohol fatalities | 13/50 | 13/51 | 0 | 0.2667 | 0.2333 | 13/50 | 0.449 |
| 359 | mutual fund predictors | 0 | 0 | 0 | 0 | 0 | 0 | 1.035 |
| 362 | human smuggling | 2/50 | 2/39 | 0 | 0.0667 | 0.0333 | 2/50 | 0.657 |
| 367 | piracy | 8/50 | 10/185 | 0.2 | 0.0667 | 0.1000 | 8/50 | 0.414 |
| 373 | encryption equipment export | 2/50 | 2/16 | 0 | 0 | 0.0667 | 2/50 | 0.703 |
| 374 | Nobel prize winners | 8/50 | 8/204 | 0 | 0.0667 | 0.2000 | 8/50 | 0.662 |
| 377 | cigar smoking | 6/50 | 6/36 | 0 | 0.1333 | 0.1333 | 6/50 | 0.958 |
| 380 | obesity medical treatment | 3/50 | 3/7 | 0 | 0.0667 | 0.0667 | 3/50 | 0.688 |
| 384 | space station moon | 6/50 | 6/51 | 0 | 0.1333 | 0.1333 | 6/50 | 1.304 |
| 385 | hybrid fuel cars | 14/50 | 14/85 | 0 | 0.1333 | 0.2667 | 14/50 | 0.85 |
| 387 | radioactive waste | 1/50 | 1/73 | 0 | 0 | 0.0333 | 1/50 | 0.417 |
| 388 | organic soil enhancement | 0/50 | 0/50 | 0 | 0 | 0 | 0/50 | 0.835 |
| 390 | orphan drugs | 6/50 | 6/122 | 0.2 | 0.0667 | 0.1333 | 6/50 | 0.367 |
| 15 | | 90/750 | 90/1241 | 0.0667 | 0.1111 | 0.1289 | 90/750 | 13.778 |

MAP (Stemming) = 0.0701

# חלק שלישי – סיכום

## א. בעיות שנתקלנו בהם –

1. האחזור לא הניב תוצאות גבוהות – שיחקנו עם הערכים של הקבועים של הפונקציות, נרמלנו את כל הגדלים שהשתשנו בהם לפי ה- maxTF של אותו ערך, ניסינו לקחת עבור השאילתות בהתחלה גם את שדה ה-desc אך ראינו שהדבר לא מקדם אותנו ובחרנו להסיר את זה מהאלגוריתם שלנו. נתנו משקל נמוך יותר למילים שחוזרות באמצעות טיפול סמנטי.
2. זמן ריצה של השאילתות – זמן הריצה של מסמך המכיל מספר שאילתות היה ארוך מדי לטעמנו, שיפרנו אותו ע"י קריאת קובץ השאילתה באמצעות מבנה נתונים שונה ומהיר יותר ב-java, וע"י ויתור על סריקת שדה ה-desc של השאילתה שגם לא החזיר תוצאות טובות יותר והתברר כמיותר ומעמיס כיון שעבור כל מילה שם היה צורך לשלוף את ערכיה מקובץ הPosting שבו היא מופיעה.
3. הגדרת הקבועים באלגוריתם BM25 – שיחקנו עם ערכי הקבועים –k1,b עד שהנבנו תוצאות משביעות רצון עבורנו. עשינו זאת בצורה אקראית ע"י מספר גדול של הרצות ובדיקת התוצאות.

## ב. האתגר הגדול בפרוייקט –

החלוקה המודולרית של המחלקות בחלק א'. האתגר שהיה מבחינתנו הכי גדול היה לחלק את המחלקות בצורה נכונה כך שלכל מחלקה יהיה את התפקיד שלה והן לא יתנגשו אחד עם השניה. את תכנון הפרוייקט במיוחד בחלק א' עשינו במשך המון המון זמן עד שגיבשנו החלטה סופית למבנה הארכיטקטורה של המחלקות ותחומי האחריות של כל אחת.
בסופו של דבר התכנון המעמיק התברר כחשוב מאוד וגם כמועיל כאשר הגענו לחלק ב' וכתבנו את כולו מבלי לשנות שום דבר מהותי פרט לדברים מזעריים מאוד בחלק א'.

## ג. המלצות לשיפור האלגוריתם שלנו –

במידה והיה לנו עוד זמן, היינו מוסיפים אלגוריתמים נוספים לחישוב הדירוג של מסמך כמו למשל – cosSim שנלמד בכיתה.
בנוסף היינו נותנים ציון שונה למסמכים שמילים מהשאילתה מופיעות בכותרת המסמך.
דבר נוסף שהיינו נותנים ציון שונה לפיו הוא שימוש ברשימת ה-5 יישויות הדומיננטיות שמצאנו עבור כל מסמך, במידה והיישות היתה מופיעה בשאילתה היינו מדרגים את אותו מסמך משמעותית יותר מאחרים.
לצערנו לא היה לנו מספיק זמן להשתמש בכל מה שתכננו ורצינו ולהגיע לתוצאות טובות יותר, אך מבחינת הבניה של הקבצים דאגנו ליצור אותם כך שכל הדברים שתכננו לעשות על מנת לשפר את האלגוריתם שלנו מחושבים ונמצאים מבלי לעשות שום שינוי בבניית הקבצים אלא רק לעשות בהם שימוש.