מגישות : עמית שקרצ'י 313278889

שירן גולזר 205866106

אחזור מידע

דו"ח פרויקט- מנוע חיפוש חלק 2

1. **עיצוב התכנה:**

a. בסעיף זה, נתאר באופן מפורט את אופן פעולת התכנה, את מחלקותיה ואת השיטות של כל מחלקה.

תיאור פעולת התכנה- "מבט על":

בשלב זה של העבודה, כבר יש בידינו מילון תקין ומלא. המערכת מקבלת מהמשתמש שאילתא בודדת (או קובץ שאילתות) לאחזור המסמכים. לאחר מכן, המערכת מעבירה את השאילתא למחלקת Searcher , שם מתבצע פרסור של השאילתא ל- tokens כפי שנעשה הפרסור של המילון עצמו.
לאחר פרסור השאילתא, המערכת מאחזרת את רשימת המסמכים לכל term בשאילתא מתוך קבצי ה- posting, ומעבירה את כלל המידע למחלקת ה- Ranker לשלב דירוג התוצאות.
בסיום הדירוג, מוצגים למשתמש 50 המסמכים הרלוונטיים ביותר שנמצאו עבור השאילתא, בתוספת אפשרות להצגת חמש הישויות הדומיננטיות במסמך.

b. בסעיף זה נתאר את המחלקות והשינויים במחלקות.

תיאור מחלקות התכנה:

1. **מחלקת Searcher-**
המחלקה מקבלת שאילתא, מפרסרת אותה ומחלצת את רשימת המסמכים הרלוונטיים מתוך קבצי ה- posting של המילון.

תיאור שדות המחלקה:

שדה	תיאור
toStem	משתנה בוליאני המחזיק את רצון המשתמש- האם בוצע stemming או לא
useSemanticModel	משתנה בוליאני המחזיק את רצון המשתמש – האם לאחזר תוצאות בעזרת חיפוש סמנטי או לא (באמצעות word2vec)
useDataMuseAPI	משתנה בוליאני המחזיק את רצון המשתמש – האם לאחזר תוצאות בעזרת חיפוש סמנטי ע"י התחברות לממשק API אינטרנטי.
dirPostingDic	מחזיק את הניתוב בו נמצא המילון וקבצי ה- posting
parser	מטרתו לפרסר את השאילתא עצמה
dictionary	המילון.
docsQuery	מחזיק עבור כל שאילתא את המסמכים שחזרו

תיאור הפונקציות:

- בנאי- מקבל את המילון ותיקיית dirPostingDic.
- setToStem- מעדכנת את שדה ה-toStem של ה- מחלקה.
- setDataMuseAPI- מעדכן את שדה ה-useDataMuseAPI של ה- מחלקה.
- setUseSemanticModel- מעדכן את שדה ה-useSemanticModel של ה- מחלקה.

- search- הפונקציה מקבלת שאילתא, ומחזירה רשימת מסמכים רלוונטיים אליה, בצירוף ה- terms מהשאילתא לכל מסמך.
- Clear- הפונקציה מוחקת את כל מבני הנתונים מהזיכרון הראשי.
- searchSemantic- הפונקציה מקבלת שאילתא ומחזירה רשימת מסמכים בצירוף המלים הדומות סמנטית למילות השאילתא, בהתאמה.
- getDocsFromPosting- השאילתא מקבלת term מתוך השאילתא, ונתיב לקובץ הפוסטינג ומיקום בקובץ, ומחזירה את רשימת המסמכים בהם הוא מופיע.
- textOperations- הפונקציה מקבלת שאילתא ומחזירה פרסור שלה.
- getSemanticWord- מקבלת שאילתא מפורסרת ומחזירה רשימת מלים דומות סמנטית עבור על term בשאילתא.

2. **<u>מחלקת Ranker-</u>**

המחלקה מקבלת אוסף מסמכים ואת השאילתא המפורסרת, ומדרגת תוצאות החיפוש.

<u>תיאור שדות המחלקה:</u>

שדה	תיאור
toStem	משתנה בוליאני המחזיק את רצון המשתמש- האם בוצע stemming או לא
dictionary	המילון.
B	ערך קבוע בפונקציית הדירוג
K	ערך קבוע בפונקציית הדירוג
AVERAGE_LENGTH	ערך קבוע בפונקציית הדירוג
docsDictionary	מחזיק את מילון המסמכים.
docsEntities	מחזיק את מילון המסמכים והישויות המופיעות בו.

<u>תיאור הפונקציות:</u>

- בנאי- מקבל את המילון, את מילון המסמכים, ואת מילון הישויות.
- setToStem- מעדכנת את שדה ה-toStem של ה- מחלקה.
- Rank- הפונקציה מקבלת את רשימת המסמכים לדירוג, ומחזירה את דירוג המסמכים בצורה ממוינת מהרלוונטי ביותר לרלוונטי פחות.
- getBM25- פונקציית מעטפת המקבלת רשימת מסמכים ומחזירה דירוג של אוסף המסמכים לפי נוסחת BM25.
- getBM25- הפונקציה הפנימית. מקבלת מסמך בודד ומחזירה את תוצאות החישוב של נוסחת BM25 עבור המסמך הנתון.
- getIDF- הפונקציה מקבלת term ומחזירה את ערך ה- IDF שלו.
- getTF- הפונקציה מקבלת מסמך ו- term. ומחזירה את ערך ה- TF של ה-term במסמך הנתון.
- rankWithWeight- מקבלת שאילתא שתי רשימות מסמכים ומדרגת אותם לפי משקלים.
- getTopEntities- הפונקציה מקבלת docID ומחזירה את 5 הישויות הדומיננטיות והציון של כל אחת מהן, עבור אותו מסמך.
- Clear- הפונקציה מוחקת את כל מבני הנתונים מהזיכרון הראשי.

3. **מחלקת diskWriter-**
המחלקה אחראית על כתיבת מבני הנתונים הקשורים למילון לדיסק.

תיאור הפונקציות:
- writeDocsEntitiesToDisk- הפונקציה כותבת לדיסק את מילון הישויות.
- writeDictionaryToDisk - הפונקציה כותבת את המילון לדיסק.
- writeResultsToDisk- הפונקציה כותבת את תוצאות השאילתא לקובץ.

4. **מחלקת diskReader-**
המחלקה אחראית על כתיבת מבני הנתונים הקשורים למילון לדיסק.

תיאור הפונקציות:
- readDictionaryFromDisk- הפונקציה מעלה את המילון לזיכרון הראשי.
- readDocsEntitiesFromDisk- הפונקציה מעלה את מילון הישויות לזיכרון.

5. **מחלקת DataMuse-**
המחלקה אחראית להתחברות עם ממשק DataMuse API, ולמציאת ביטויים דומים סמנטית.

תיאור הפונקציות:
- בנאי- מאתחל את שדות המחלקה
- findSynonyms- הפונקציה מקבלת term, ומחזירה terms נוספים הדומים לו סמנטית.
- getArr- הפונקציה מפרקת את תוצאות החיפוש למערך ומחזירה אותו.

6. **מחלקת Model-**
- סימנו בכחול חלקים שלא שונו מחלק א של העבודה.

המחלקה אחראית לקישור בין ה- Controller לגוף המערכת.

תיאור שדות המחלקה:

שדה	תיאור
toStem	משתנה בוליאני המחזיק את רצון המשתמש- לבצע stemming או לא
readFile	מחזיק את אובייקט מסוג ReadFile
dictionary	מחזיק את המילון
docsDictionary	מחזיק את מילון המסמכים
useSemanticModel	משתנה בוליאני המחזיק את רצון המשתמש – האם לאחזר תוצאות בעזרת חיפוש סמנטי או לא (באמצעות word2vec)
useDataMuseAPI	משתנה בוליאני המחזיק את רצון המשתמש – האם לאחזר תוצאות בעזרת חיפוש סמנטי ע"י התחברות לממשק API אינטרנטי.
DirPostingsDic	מחזיק את הניתוב בו נמצא המילון וקבצי ה- posting
docsEntities	מחזיק את מילון המסמכים והישויות המופיעות בו.
docsRank	מחזיק את דירוג המסמכים עבור שאילתא בודדת

תיאור הפונקציות:
- בנאי- מאתחל את שדות המחלקה.
- getDictionary - מחזירה את המילון.

- getDocsDictionarySize - הפונקציה מחזירה את מספר המסמכים שאונדקסו.
- getDictionarySize- הפונקציה מחזירה את מספר המילים הייחודיות בקורפוס (גודל המילון).
- loadDic- הפונקציה טוענת את המילון לזיכרון הראשי.
- generateDictionary- הפונקציה מקבלת ניתוב לקורפוס ובונה את המילון.
- clearAllData- הפונקציה מוחקת את כל מבני הנתונים בזיכרון הראשי, קבצי ה-posting והמילון.
- getDocsRank- הפונקציה מחזירה את דירוג המסמכים עבור שאילתא בודדת.
- getTopEntities- הפונקציה מחזירה את הישויות הדומיננטיות ביותר עבור מסמך, ואת הציון של כל ישות.
- setPostingDic- הפונקציה מעדכנת את הניתוב בו נמצא המילון וקבצי ה- posting.
- runQueriesFile- הפונקציה מקבלת קובץ עם שאילתות להרצה, ומריץ חיפוש על כל שאילתא בקובץ.
- runSingleQuery- הפונקציה מקבלת שאילתא בודדת ומריצה חיפוש עליה.
- startRanker- מאתחלת את ה- ranker כמופע בודד.
- startSearcher- מאתחלת את ה- searcher כמופע בודד.

7. **מחלקת Controller-**

- סימנו בכחול חלקים שלא שונו מחלק א של העבודה.

המחלקה אחראית לממשק המשתמש. היא מקשרת בין בקשות המשתמש לפונקציונאליות של מחלקת ה- Model ולתצוגת המערכת על המסך.

תיאור שדות המחלקה:

שדה	תיאור
mainStage	מחזיק מצביע ל- stage
browseDirPostingsDic	מחזיק מצביע לתיקייה בה נמצאים קבצי ה- posting והמילון שהמשתמש הזין
browseDirCorpusStopWords	מחזיק מצביע לתיקייה בה נמצאים הקורפוס וה-stop words שהמשתמש הזין
corpusStopWordsPath	מצביע ל- TextField המחזיק ניתוב לתיקייה בה נמצאים הקורפוס וה-stop words
postingDicPath	מצביע ל- TextField המחזיק ניתוב לתיקייה בה נמצאים המילון וקבצי ה- posting
corpusStopWordsStr	string המחזיק ניתוב לתיקייה בה נמצאים הקורפוס וה-stop words
postingDicStr	string המחזיק ניתוב לתיקייה בה נמצאים המילון וקבצי ה-posting
myModel	אובייקט המחזיק את המודל הראשי
queriesFile	מחזיק מצביע לקובץ השאילתות לחיפוש
queryTextFiled	מצביע ל-TextField המחזיק את השאילתא שהמשתמש הזין
browseDirResults	מחזיק מצביע לתיקייה בה נמצאות תוצאות החיפוש, שהמשתמש הזין
loadDictionary	משתנה בוליאני המחזיק האם המילון טעון או לא.

תיאור הפונקציות:

- initialize- הפונקציה מאתחלת את שדות המחלקה.
- startButton- הפונקציה מופעלת בעת לחיצה על כפתור - הstart וקוראת למודל לבניית המילון.
- loadDictionary- הפונקציה טוענת את המילון לזיכרון הראשי.
- stemmingCheckBox- הפונקציה מעדכנת האם לבצע stemming בהתאם לבחירת המשתמש.
- browseDirPostingsDictionary- הפונקציה מעדכנת את ניתוב browseDirPostingsDic.

- browseDirCorpuseStopWords- הפונקציה מעדכנת את ניתוב browseDirCorpusStopWords.
- restart- הפונקציה מוחקת את כל קבצי ה- posting והמילון ומוחקת את כל הזיכרון הראשי של התוכנית.
- displayDictionary- הפונקציה מציגה את המילון בחלון חדש.
- informationAlert- הפונקציה מקבלת הודעה ומציגה אותה למסך.
- browseQueriesButton- הפונקציה מעדכנת את ניתוב הקובץ של השאילתות להרצה.
- browseQueryResultsPath- הפונקציה מעדכנת את הניתוב לשמירת תוצאות הריצה.
- semanticModelCheckBox- הפונקציה מעדכנת האם לבצע חיפוש סמנטי ע"פ דרישת המשתמש.
- runSearchButton- הפונקציה מתחילה את חיפוש השאילתה הבודדת או קובץ השאילתות בהתאמה.
- runSearchSingleQuery- הפונקציה מקבלת שאילתא בודדת ומריצה אותה.
- runSearchQueriesFile- הפונקציה מקבלת אוסף שאילתות ומריצה אותן.
- displayResults- הפונקציה מציגה למשתמש את תוצאות השאילתא בטבלה.
- addButtonToTable- הפונקציה יוצרת כפתור להוספה לטבלה.
- displayDocs- הפונקציה מציגה את המסמכים שחזרו עבור השאילתא.
- addEntityButtonToTable- הפונקציה מוסיפה כפתור "הצג ישויות" לטבלה המתאימה.
- displayEntitiesForDoc- הפונקציה מציגה את הישויות הקשורות למסמך מסוים, לבקשת המשתמש.
- setNewWindow- הפונקציה מקבלת טבלה ומציגה אותה למשתמש.
- useDataMuseAPI- הפונקציה מעדכנת האם לבצע שימוש בממשק ה-API לחיפוש סמנטי או לא.

<u>חלק שני- אלגוריתמי הדירוג:</u>

c. בסעיף זה, נסביר באופן מפורט את האלגוריתמים הכלולים במנוע.

א. <u>אלגוריתם הדירוג:</u>

$$score(Q,D) = scoreBM25(Q,D) - n_{d,q}$$

$n_{d,q}$- מספר המילים בשאילתה שמופיעות במסמך.

נוסחת ה-BM25:

$$scoreBM25(Q,D) = \sum IDF * \frac{(f(q_i,D)*(k+1))}{f(q_i,D)+k*(1-b+b*\frac{|D|}{avgdl})}$$

$|D|$- אורך המסמך D- ערך זה שמור במבנה נתונים מיוחד המכיל מידע על המסמכים עצמם.

$avgdl$- אורך ממוצע של מסמך בקורפוס. ערך זה מחושב פעם אחת, ונשמר במשתנה.

$f(q_i,D)$- מס' המופעים של המילה ה- i במסמך D- ערך זה שמור בקבצי ה- posting.

כאשר:

$$IDF = \log\left(\frac{N-df+0.5}{df+0.5}\right)$$

N- מס' המסמכים בקורפוס- ערך זה חושב פעם אחת, ונשמר במשתנה.

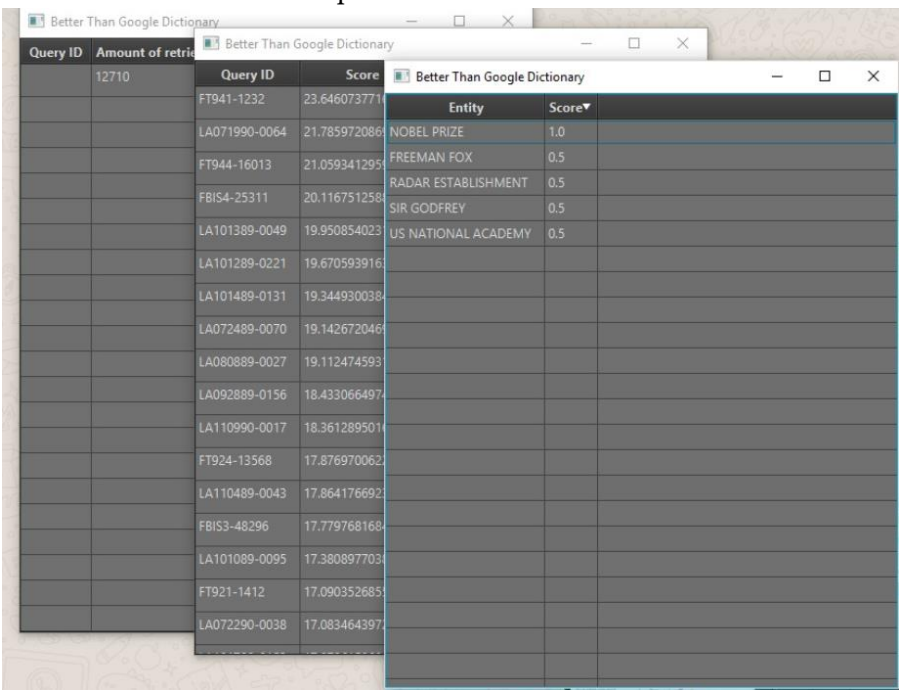
df- מס' המופעים של המילה ה- i בקורפוס. ערך זה שמור במילון עבור כל term.

ערכי הפרמטרים k,b נבחרו באמצעות ניסויים חוזרים למציאת מס' מסמכים רלוונטיים חוזרים גדול ככל הניתן.
ערכי הפרטמרים שנבחרו שונים עבור ביצוע השאילתא עם stemming או בלי.
עבור שאילתא עם stemming :
B=0.01, k=1.2
עבור שאילתא ללא stemming :
B=0.5, k=0.8

- אחד מהניסיונות לשיפור אחזור המסמכים הרלוונטיים היה לקחת גם את תגית הDISCRIPTION שהופיע בשאילתה ולהוסיף אותו כחלק מהשאילתה על מנת לאחזר מסמכים נוספים. אך ראינו שפעולה זו מקטינה את כמות המסמכים הרלוונטיים שמוחזרים ולכן הוחלט לשלוף לפי מילות השאילתה בלבד.
- ניסיון נוסף היה להתייחס בנוסחת הדירוג לכמות המילים בשאילתה המופיעות במסמך ולתת על סמך זה ציון למסמך. תחילה ניסינו לעלות לציון שחזר מנוסחת הBM25 את הציון בתוספת כמות המילים שהופיעו במסמך אך ראינו שדווקא הורדה של כמות המילים מציון המסמך מחזירה יותר מסמכים רלוונטיים ולכן הנוסחה הנ"ל נבחרה.

<u>דוגמא להרצת האלגוריתם :</u>

בדוגמא הנ"ל חיפשנו לפי השאילתה : "Nobel prize winners"



ב. <u>אלגוריתם מציאת הישויות:</u>

- יצרנו מילון מסמכים המחזיק את הישויות ותדירותן עבור כל מסמך (תוספת לשלב אינדוקס המסמכים).
- לכל מסמך רלוונטי לשאילתא, שלפנו את רשימת הישויות שלו, מיינו אותן ע"פ תדירותן במסמך.
- כל ישות במסמך קיבלה ציון ע"פ הנוסחה הבאה :

$$score(E_{ij}) = \frac{f(E_{ij})}{\max\{(E_{ij})\}}$$

E_{ij} - ישות i במסמך j.

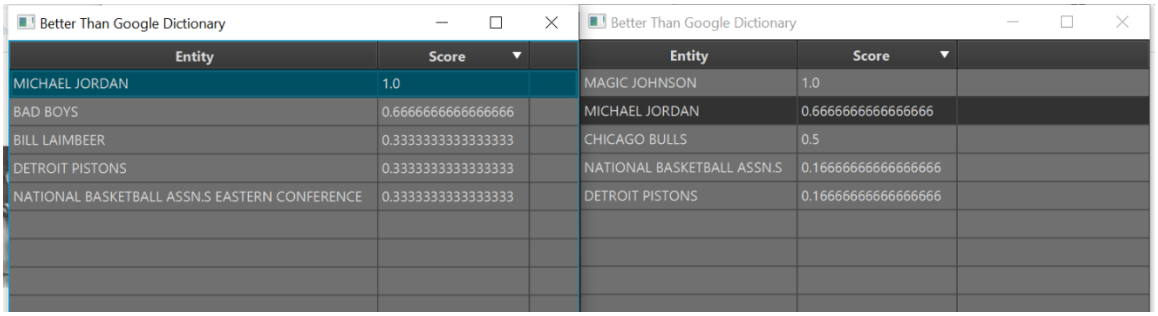
$f(E_{ij})$ - מס' המופעים של ישות i במסמך j.

$\max\{(E_{ij})\}$ – מס' המופעים של הישות שמופיעה הכי הרבה במסמך.

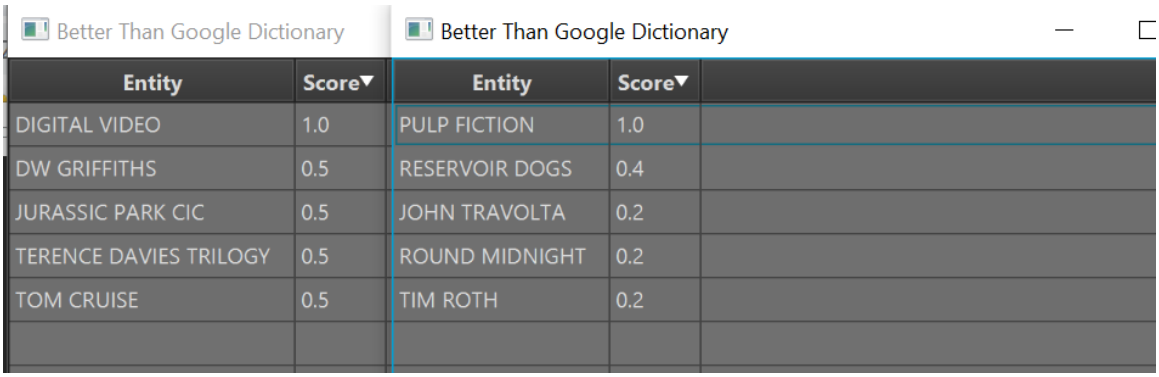
כך, ישות דומיננטית מאוד במסמך תקבל דירוג גבוה יותר מאשר ישות דומיננטית פחות.

<u>דוגמאות להרצת אלגוריתם דירוג ישויות:</u>

- <u>דוגמא 1:</u>
להלן הישויות הדומיננטיות עבור 2 המסמכים המדורגים במקומות הראשונים, שהשאילתא Michael Jorden החזירה. ניתן לראות כי כל היישויות שאותרו תואמות את הקונטקסט של החיפוש, ואף מושא החיפוש אותר כישות- עם הציון הגבוה ביותר, תואם לכך שזו הישות הדומיננטית עבור החיפוש הנ"ל.



- <u>דוגמא 2:</u>
להלן הישויות הדומיננטיות עבור 2 המסמכים המדורגים במקומות הראשונים, שהשאילתא pulp fiction החזירה. ניתן לראות כי הביטוי עצמו מופיע בתוצאות, וכן גם "כלבי אשמורה"- סרט נוסף של קוונטין טרנטינו, ואף מס' שמות של סרטים נוספים.



ג. <u>אלגוריתם לשיפור סמנטי:</u>

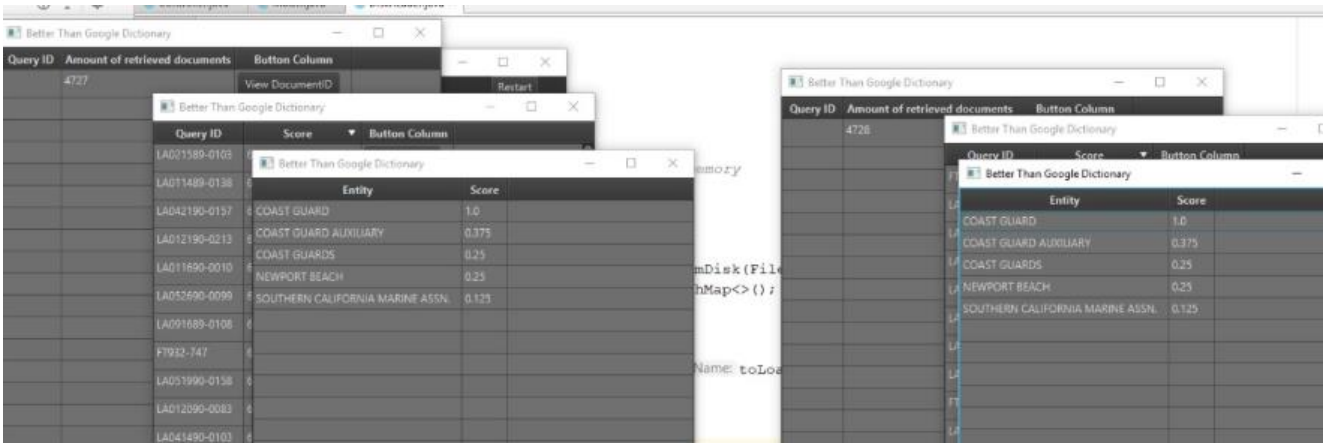
- במהלך ההחלטה לקביעת הציונים של המסמכים יחד עם המודל הסמנטי ביצענו מספר ניסיונות של שליפות על מנת לבדוק איזה מבין הניסיונות יניב תוצאות טובות יותר. אחד מהניסיונות היה להוסיף את המילים שהתקבלו כנרדפות למילות השאילתה כחלק מהשאילתה (query expansion) ובכך לתת משקל שווה לכל המילים (השאילתה והנרדפות). אך, ראינו ששימוש כזה הוא פחות יעיל מאשר ציון לפי חלוקה למשקלים.

a. <u>שימוש ב-API חיצוני עבור מציאת מלים נרדפות ודומות סמנטית:</u>
ממשק ה-Datmuse API מספק אפשרות למציאת מלים נרדפות או דומות למילה מסוימת.
לפני שלב החיפוש נבצע את הפעולות הבאות:
עבור כל מילה מתוך מילות השאילתא, נמצא בעזרת ה- API את המלים הדומות לה. נבנה שאילתא המורכבת מהמלים הנרדפות, ונמצא את הערכים שלה. כעת נתחיל בשלב החיפוש.
בשלב החיפוש, נבנה רשימה של המילים הנרדפות שמצאנו ולהן את המסמכים והתדירות שלהם בכל מסמך. ומשם נעבור לשלב הדירוג.
בשלב הדירוג, נחשב את הציון למסמך כש0.1 מהציון יתייחס למשקל שקיבל המסמך מהמילים הנרדפות ו0.9 מהציון יתייחס למשקל שיתקבל ממילות השאילתה המקורית.
המשקלים חולקו במחשבה שהמילים הנרדפות אינן המילים שהמשתמש בחר לשלוף ואף יכולות להציג מסמכים לא רלוונטיים.
חשוב לציין כי על מנת להשתמש באופציה זו ראשית יש להתחבר לאינטרנט.

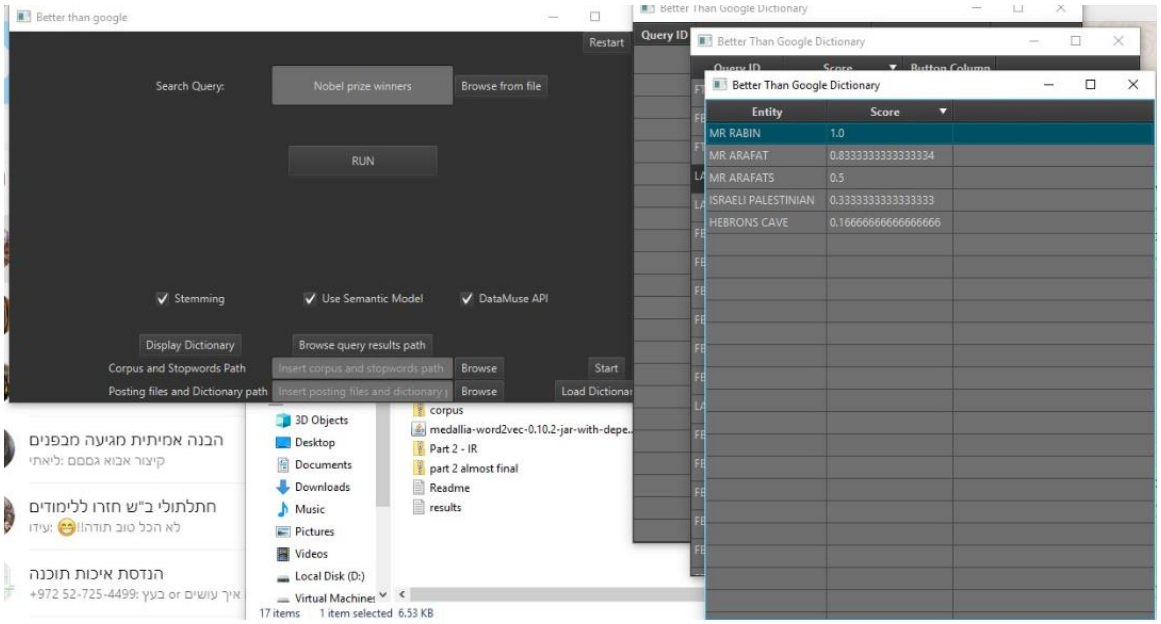
b. <u>שימוש בספריית Medalia כדי לממש word2vec:</u>
לכל מילה ממילות השאילתה, נמצא מילה אחת דומה סמנטית בעזרת הספריה הנ"ל. נבנה שאילתא המורכבת מהמלים הנרדפות, ונמצא את הערכים שלה. כעת נתחיל בשלב החיפוש.
בשלב החיפוש, נבנה רשימה של המילים הנרדפות שמצאנו ולהן את המסמכים והתדירות שלהם בכל מסמך. ומשם נעבור לשלב הדירוג.
בשלב הדירוג, נחשב את הציון למסמך כש0.1 מהציון יתייחס למשקל שקיבל המסמך מהמילים הנרדפות ו0.9 מהציון יתייחס למשקל שיתקבל ממילות השאילתא המקורית.
המשקלים חולקו במחשבה שהמילים הנרדפות אינן המילים שהמשתמש בחר לשלוף ואף יכולות להציג מסמכים לא רלוונטיים.

<u>דוגמאות להרצה עם שימוש במודל הסמנטי:</u>

- <u>דוגמא 1:</u>
בדוגמא הנ"ל, חיפשנו את המילה boat עם ובלי שימוש ב- סמנטיקה.
המסמכים עבורם אנו רואים את הישויות, קיבלו דירוג שונה בהתאם לשימוש באחזור עם סמנטיקה- נראה כי כאשר אחזרנו ללא השימוש בממשק קיבלנו את המסמך "LA021589-0103" במקום החמישי לעומת זאת, עם שימוש בממשק קיבל המסמך את המקום הראשון.
לפי דירוג הישויות אפשר להבין שמאחר והמסמך מכיל ישויות דומות מבחינה סמנטית לboat אז המסמך קיבל ציון גבוה יותר (לדוגמא coast guard).



- <u>דוגמא 2:</u>
בדוגמא הנ"ל ביצענו חיפוש לשאילתה "Nobel prize winners" באמצעות שימוש במודל סמנטי וללא שימוש. ראינו שכאשר לא השתמשנו במודל קיבלנו תוצאות שנוגעות יותר למונח עצמו ופחות למשמעות שלו ואילו עבור שימוש במודל הסמנטי קיבלנו דירוג שונה של מסמכים עם ישויות שקשורות לשאילתה לדוגמא: "MR RABIN" ו"MR ARAFAT"



d. בסעיף זה נסביר על נתוני קבצי הpostingים המילונים ששמרנו לאחזור.

- <u>קובץ הpostingים:</u> מכיל את הtermים ורשימת מסמכים (docID) וTF של כל term במסמך. נתון הTF עוזר בחישוב הBM25 למתן ציון של המסמכים.
- <u>מילון:</u> המילון מכיל את הניתוב לקובץ הpostingים עבור כל term ואת המיקום שלו בקובץ דבר שמזרז את החיפוש של הtermים בקובץ הpostingים ואת זמן האחזור. בנוסף המילון מחזיק את הdf של כל term נתון שהשתמשנו בו לחישוב מתן הציון למסמך בנוסחה של BM25.
- <u>מילון ישויות ומסמכים:</u> יצרנו מילון של מסמכים והישויות שלהם בצירוף תדירות הישויות במסמך. מילון זה עזר לנו לשלוף בקלות את הישויות של אותו מסמך ולדרג את ה5 הכי דומיננטיות במסמך.

e. בסעיף זה נציג את המקורות לשימוש בקוד הפתוח (פירוט השימוש הוא בסעיף "אלגוריתם לשיפור סמנטי")

- <u>שימוש בספריית Medalia כדי לממש word2vec:</u>
  * medallia-word2vec-0.10.2-jar-with-dependencies jar
  * word2vec.c.output.model.txt
- <u>שימוש ב-API חיצוני:</u>
  * https://www.datamuse.com/api/

2. **<u>הערכה של המנוע:</u>**

<u>ללא שימוש ב- stemming:</u>

Query ID	Query	Precision	Recall	Precision@5	Precision@15	Precision@30	Precision@50	Runtime (milliseconds)
351	Falkland petroleum exploration	0.3	0.3125	0.4	0.4	0.3	0.3	407
352	British Chunnel impact	0.28	0.056	0.4	0.4667	0.333	0.28	470
358	blood-alcohol fatalities	0.26	0.25	0.2	0.4667	0.3	0.26	416
359	mutual fund predictors	0.06	0.1	0	0.1333	0.1	0.06	1379
362	human smuggling	0.18	0.23	0.4	0.2667	0.233	0.18	691
367	piracy	0.156	0.4	0	0	0.0333	0.156	176
373	encryption equipment export	0.12	0.375	0	0.2	0.2	.12	768
374	Nobel prize winners	0.58	0.142	1	0.7333	0.6667	0.58	965
377	cigar smoking	0.22	0.305	0	0.0667	0.1333	0.22	752
380	obesity medical treatment	0.1	0.714	0	0.2667	0.1333	0.1	487
384	space station moon	0.2	0.19	0.6	0.2667	0.2	0.2	1155
385	hybrid fuel cars	0.38	02223	0.4	0.4	0.4	0.38	481
387	radioactive waste	0.14	0.095	0	0.1333	0.1333	0.14	44
388	organic soil enhancement	0.16	0.16	0	0.1333	0.1667	0.16	549
390	orphan drugs	0.22	0.09	0.2	0.2	0.3	0.22	408

Map (average precision over all rel docs) = 0.0685
Total Precision= 0.224
Total recall=0.135

עם שימוש ב- stemming:

Query ID	Query	Precision	Recall	Precision@5	Precision@15	Precision@30	Precision@50	Runtime (milliseconds)
351	Falkland petroleum exploration	0.38	0.395	0.2	0.5333	0.4667	0.38	9436
352	British Chunnel impact	0.24	0.048	0.2	0.4	0.333	.24	806
358	blood-alcohol fatalities	0.3	0.29	0.2	0.3333	0.2333	0.3	191
359	mutual fund predictors	0.06	0.107	0	0.1333	0.0667	0.06	1204
362	human smuggling	0.12	0.154	0	0.0667	0.1333	0.12	638
367	piracy	0.16	0.043	0	0	0.0667	0.16	160
373	encryption equipment export	0.12	0.375	0	0.2667	0.2	0.12	679
374	Nobel prize winners	0.58	0.142	1	0.6	0.5667	0.58	675
377	cigar smoking	0.26	0.36	0	0.1333	0.1333	0.26	640
380	obesity medical treatment	0.1	0.7	0.2	0.2667	0.1333	0.1	375
384	space station moon	0.2	0.196	0.2	0.2	1.333	0.2	1472
385	hybrid fuel cars	0.42	0.247	0	0.3333	0.4667	0.42	680
387	radioactive waste	0.22	0.15	0.2	0.1333	0.11	0.22	52
388	organic soil enhancement	0.1	0.1	0.2	0.1333	0.1	0.1	947
390	orphan drugs	0.26	0.1065	0.2	0.2667	0.333	0.26	422

Map (average precision over all rel docs) = 0.0633
Total Precision= 0.235
Total recall=0.142

3. **סיכום:**

במהלך העבודה נתקלנו עם מס׳ בעיות. נפרט כיצד התמודדנו עמן:

a. בעית מקום בזיכרון הראשי- היה עלינו להתמודד עם כמויות עצומות של מידע במהלך עבודת העיבוד של ה- corpus. התמודדנו עם הבעיה באמצעות פיצולה למנות קטנות- עבדנו כל פעם על גודל מסוים של קבוצת מסמכים, כדי למנוע את התפוצצות המקום הפנוי ב- RAM.

b. בעית זמן ריצה- היה עלינו להתמודד מול זמני ריצה ארוכים ולקצרם. במהלך החלק הראשון של העבודה, בניית המילון, ביצענו מספר אופטימיזציות לקיצור זמני הריצה (הכולל התייחסות למבני נתונים כמו HASH). בחלק השני של העבודה תכננו את שליפת הנתונים כך שזמן הריצה של החיפוש יהיה קצר ככל הניתן.

c. בניית נוסחת הדירוג, עם משקולות עבור כל חלק בדירוג- ביצענו ניסויים רבים, והעלנו השערות רבות הנוגעות לדרך הטובה ביותר לשיפור תוצאות השאילתה.

d. עיבוד המידע- אתגר נוסף בפרויקט היה לעבד מידע גדול (הcorpus) ולסנן רק את המידע הרלוונטי עבורנו. כלומר, לפרסר את המסמכים לterms נקיים.

המלצות לשיפור האלגוריתם שלנו:

a. מציאת ספריה הממשמשת מודל סמנטי- המימוש בו השתמשנו עבור word2vec אינו מספק תוצאות מאוד איכותיות, ולכן לא נתנו משקל גדול לציון עבור המילים שהמודל החזיר. מימוש טוב יותר למודל הסמנטי, היה מביא שיפור משמעותי בתוצאות.

b. הוספת פיצ׳רים נוספים לדירוג השאילתא- שימוש בתאריך הוצאת המסמך עבור דירוג הרלוונטיות שלו, יכולה להיות תוספת טובה.