מגישים: איתי כץ 312178312
תומר זיינפלד 312543929

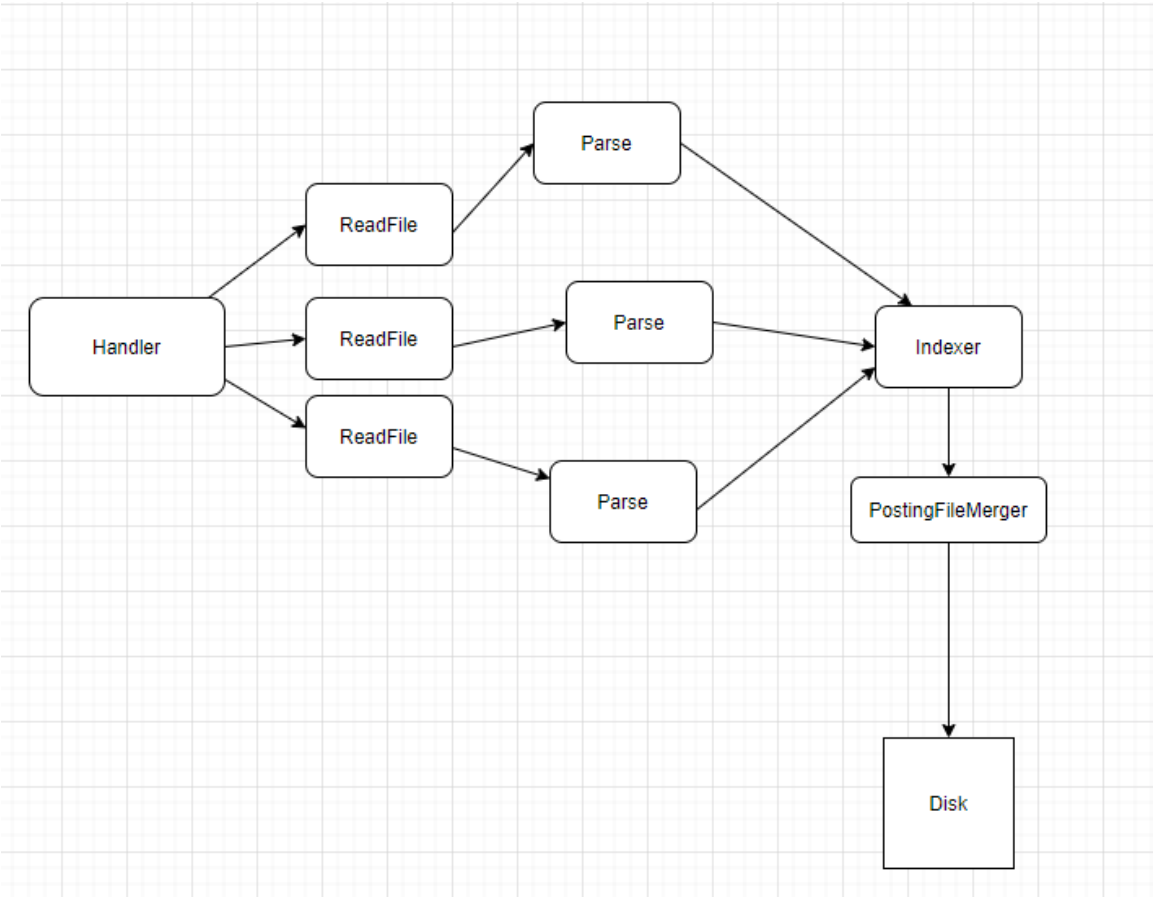
דו"ח מנוע אחזור – חלק ב'

אופן פעולת המנוע

שלב בתהליך	אובייקט	תיאור
לאורך כל התהליך	Handler	אובייקט העל אשר מקושר לכל האובייקטים בתהליך האחזור. האובייקט מתחיל את התהליך, ואחראי על קבלת הפלטים הסופיים והעברתם ל-GUI.
1	ReadFile	אחראי על קריאת התיקיות שמרכיבות את ה-Corpus, והעברתן לתוך ה-RAM כמסמכים.
2	Parse	אחראי על קבלת מסמך, ופיצולו לאבני היסוד של המילון, שהן ה-Term-ים. בשלב זה, הוא עוטף את המילים עם המידע הרלוונטי, לאחר סיווגן לקטגוריות, והעברתן כאובייקט Term להמשך התהליך.
3	DocWriter	אובייקט האחראי על כתיבת המידע על המסמכים לדיסק, וכן את רשימת המילים הפוטנציאליות להיות ישויות בכל מסמך.
4	Indexer	אובייקט האחראי על בניית המילון. האובייקט מקבל מבנה נתונים המחזיק את ה-Term-ים, ומהם הוא יוצר מילון, אשר מכיל מידע על כל מילה, בין היתר על מספר המסמכים שבהן היא מופיעה, מספר המופעים בכל מסמך, האם עליה להישמר באות גדולה או קטנה ועוד.
5 (אופציונלי)	Stemmer	מחלקה אשר לוקחת Term ומעבירה אותה תהליך Stemming בהתאם לבקשת המשתמש. האובייקט מגיע בשילוב עם ה-Indexer לפני ההכנסה למילון.
6	PostingFileMerger	אובייקט האחראי לקחת את המילון, ולכתוב אותו לדיסק בצורה של קבצי PostingFile.
7	Query	אובייקט העוטף את מילות השאילתה שהמשתמש הכניס למערכת (בין אם כקובץ או כמחרוזת עם רווחים). האובייקט מחזיק את רשימת המילים שהמשתמש הכניס, את המזהה של השאילתה ואת המחרוזת של השאילתה.

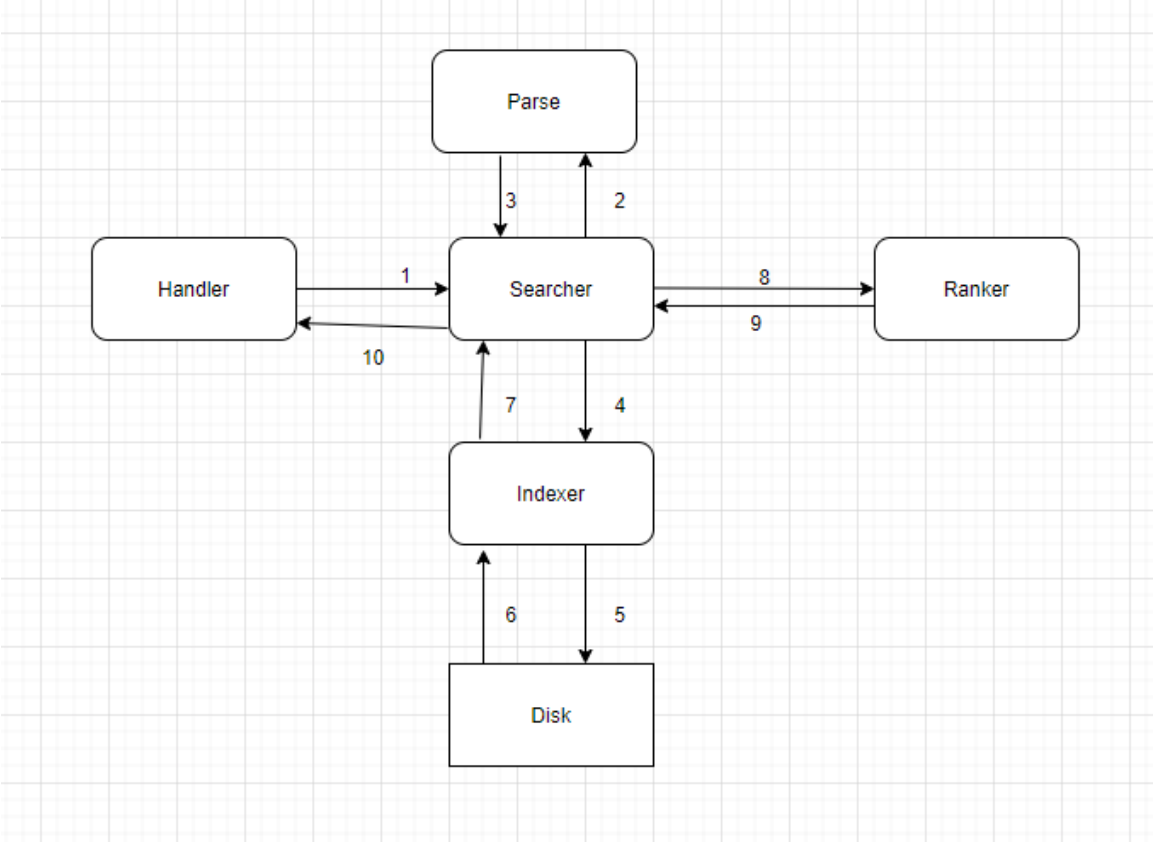
8	Searcher	אובייקט הלוקח שאילתא, ומחזיר את המסמכים המכילים את מילות השאילתא (או מילים דומות בהנחה שמשתמשים בסמנטיקה).
9	Ranker	אובייקט הלוקח את רשימת המסמכים ואת רשימת המילים אשר כל מסמך מחזיק בתוכו (מילים אשר מוכלות בתוך השאילתא) ותפקידו לדרג את המסמכים, ולאחר מכן להחזיר את המסמכים הרלוונטים ביותר.
10	QueryData	אחראי לעטוף את התוצאות אשר חוזרות מהשאילתא, כך שיהיה ניתן להציג למשתמש בצורה נוחה.

להלן תרשים אובייקטים לתהליך הפירסור של הקורפוס והפיכתו למילון:



מגישים: איתי כץ 312178312
תומר זיינפלד 312543929

להלן תרשים אובייקטים לאחזור שאילתה כאשר המספר מציין את השלב בתהליך :



הסבר מורחב על המחלקות:

מחלקות שעברו שינוי מחלק א':

המחלקה	השינוי
Parse	שיפור הפונקציה clearChar – בחלק א' המילון הכיל הרבה מילים, אך הרבה מהן נוצרו מכיוון שהן הכילו תווים מיוחדים שמצד אחד לא היתה להן שום משמעות בשפה האנגלית, אך מנגד הגדילו את המילון. על כן, שיפרנו את הפונקציה המוזכרת, בכך שכעת היא עוברת גם מתחילת המחרוזת, ולאחר מכן מסופה, ובצורה סדרתית מסירה תווים לא רצויים שאינם בשפה האנגלית או נומריים. הוספת פונקציה זו הסירה מילים רבות שאונדקסו קודם לכן במילון.
	שיפור הפונקציה parseArticles –

	בחלק א' כפי שהוזכר בסעיף מעלה, המילון הכיל הרבה מילים לא רלוונטיות לשפה האנגלית. הגענו למסקנה, כי תהליך הניקוי של המילים אינו טוב מספיק, ויש לבחון מילים שאינן מורכבות רק מאותיות או מספרים. על כן, פיצלנו את תהליך הפירסור ל-2 חלקים: בחלק הראשון, מנסים לשייך את המילה לקטגוריה כלשהי. בחלק השני, אם המילה לא התאמתה על אף אחד מהחלקים, יש להעביר אותה ב-REGEX אשר לוקח את כל התווים המיוחדים ומוחק אותם. לאחר מכן, מעבירים את המערך של המילים שנוצרו מהפיצול שוב פעם את תהליך הפירסור.
	הוספת הפונקציה ParseQuery – הפונקציה מקבלת אובייקט מסוג Query ומעבירה אותו בתהליך זהה לפירסור של Document, כדי שיהיה ניתן למצוא את המסמכים הרלוונטים לשאילתות.
Indexer	הוספת הפונקציה getDataFromPosting – פונקציה אשר מקבלת את התו הראשון של המילה המבוקשת, ואת ה-Pointer שלה מהמילון, ויודעת לשלוף את השורה הרלוונטית מה-PostingFile.
	הוספת הפונקציה clearWrongEntities – בחלק א' המילון הכיל הרבה מילים. אחת הסיבות לכך, שזיהינו בחלק ב', הייתה אופן הטיפול בישויות פוטנציאליות המתקבלות מה-Parser. לפני כתיבת הדיסק למילון, הפונקציה סורקת את מבנה הנתונים של המילון, ומזהה ישויות לפי המופע שלהן במילון באות גדולה, ומסירה אותן אם הן הופיעו רק במסמך אחד. פונקציה זו היא הגורם המצמצם המשמעותי ביותר של המילון.
Document	הוספת שדה למחלקה אשר מחזיק את כל היישויות הפוטנציאליות של המסמך. בשל הדרישה להצגה של חמש הישויות החזקות ביותר בכל מסמך, והעובדה כי לא נקבל את הקישור ל-corpus, הגענו למסקנה כי הדרך הפשוטה ביותר לממש זאת בפרויקט תהיה כתיבה לדיסק של כל המילים שסווגו על ידי ה-Parser כישויות. יש לזכור, כי כאן לא מסתיים התהליך, שכן היחיד שיכול להכריע אם מילה מסוימת היא ישות, זה ה-Indexer שמחזיק את המילון. הוספת השדה למעשה השפיעה על גודל הקובץ שמכיל את הפרטים על כל ה-documents.
ReadFile	הוספת הפונקציה readQueriesFile – הפונקציה קוראת מסמך המכיל את השאילתות
	הוספת הפונקציה copyStopWords –

	האובייקט קורא את ה-stop words עוד בשלב הפירסור, אך מכיוון שבחלק השני של העבודה הניתוב ל-corpus לא מוכנס ע"י המשתמש, עלינו להעתיק את הרשימה לניתוב אחר שנגיש לנו, על מנת שנוכל לסנן stop words שיגיעו מהשאילתות.
Handler	הוספת הפונקציות – findRelevantDocumentsForManyQueries findRelevantDocumentsForSingleQuery writeResultsOfMultiQueries findMostRankedEntities פונקציות אשר תומכות בדרישות של ה-GUI שהתווספו בחלק ב' של העבודה. הדרישות הינן : אחזור של שאילתה בודדת, אחזור של מסמך שאילתות, מציאה של חמש הישויות החזקות במסמך, וכן כתיבה של המידע מהשאילתות לדיסק.
Controller	הוספת הפונקציות – showMultipleQueriesResults searchMultipleQueries showMostRankedEntities runSingleQuery פונקציות אשר תומכות בדרישות של ה-GUI שהתווספו בחלק ב' של העבודה. הדרישות הינן : אחזור של שאילתה בודדת, אחזור של מסמך שאילתות, מציאה של חמש הישויות החזקות במסמך, וכן כתיבה של המידע מהשאילתות לדיסק.

מחלקות חדשות בחלק ב':

מחלקת Searcher

אובייקט האמון על קבלת ה-Query מהמשתמש, וחיפוש המסמכים עבור אותו Query.

להלן הפונקציות המרכזיות של האובייקט:

שם הפונקציה	פירוט
runUserQuery	פונקציה אשר מקבלת אובייקט מסוג Query, מפרקת אותו למילים, ושולחת את המילים ל-Parser. לאחר מכן, הוא מקבל בחזרה רשימה של

	מילים שעברו את תהליך הפירסור, ושולחת אותן לאובייקט מסוג Ranker עליו נפרט בהמשך הדו"ח.
getDocumentsFromDisk	פונקציה אשר קוראת מהדיסק את הקובץ אשר מכיל מידע על כל מסמך, ומכניסה אותו למבנה נתונים ב-RAM.
getSortedDocsList	פונקציה המקבלת מפה עם שם המסמך והציון שלו (את הציון המסמך מקבל ב-Ranker), וממיינת את ה-50 הרלוונטיים ביותר לפי הדירוג.

מחלקת Ranker –

אובייקט האמון על ביצוע הדירוג של המסמכים. החישוב מבוצע באמצעות התייחסות לנתונים מה-Posting File עבור כל מילה בשאילתא שמופיעה במסמך. האובייקט משתמש במספר אלגוריתמי דירוג, ומתן משקל לכל אחד מהם. בסוף התהליך, הוא מחזיר מסמך ואת הדירוג שלו.

שם הפונקציה	פירוט
rankRelevantDocuments	פונקציה המקבלת רשימה של Terms המרכיבים את ה-Query, ומחזירה HashMap של מסמך והדירוג שלו.
readPotentialEntitiesFromDisk	קוראת מהדיסק את כל הישויות הפוטנציאליות שיש למסמך. זאת עבור הצגה של ה-5 ישויות החזקות במסמך.
getSortedEntitiesList	פונקציה המקבלת רשימה של כל הישויות במסמך, ומדרגת אותן לפי נוסחת TF-IDF
calculateTF calculateIDF	תתי פונקציות המחשבות את משקל הישות ביחס לכל הקורפוס, המכפלה של תוצאות הפונקציות הופך להיות הדירוג הסופי של כל ישות.
removeWrongEntities	פונקציה אשר לוקחת את רשימת הישויות הפוטנציאליות מהדיסק ובודקת אותן מול המילון. בהימצא שישות מסוימת מופיעה במילון, היא

	מוסיפה אותה למבנה נתונים של ישויות שידורג בהמשך.

מחלקת Query

מחלקה המייצגת שאילתה של משתמש. מטרתה לעטוף את המחרוזת של השאילתה בפרטים נוספים, כדי שיהיה ניתן להציג אותם בהמשך למשתמש.

אין פונקציות מרכזיות במחלקה.

מחלקת QueryData

מחלקה המייצגת תצוגה של Table View ב-GUI. מטרת המחלקה היא מתן אפשרות הצגה נוחה למשתמש של תוצאות השאילתה אותה הוא הריץ.

אין פונקציות מרכזיות במחלקה.

אלגוריתמים במנוע

אלגוריתם הדירוג

אלגוריתם הדירוג שלנו מורכב משני חישובים:

חלק א' – BM25

מימשנו בקוד את הנוסחה של BM25, כאשר אנו עוברים על כל מסמך ומחשבים את הדירוג שלו בהתאם למילים בשאילתה שהוא מכיל.

להלן הנוסחה:

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

****חשוב לציין:** כאשר המשתמש בוחר להשתמש בסמנטיקה לצורך אחזור, השימוש בפונקציה משתנה בצורה הבאה: כאשר המילה שמאומתת על ידי הנוסחה מזוהה כמילה מקורית של השאילתה, היא נסכמת במשקלה המקורי, אך כאשר המילה מזוהה כמילה סמנטית המשקל שמתווסף למשקל הכולל מנורמל לכדי רבע מהמשקל המקורי שניתן.

חלק ב' – דירוג אישי שלנו – נסמן כ-PersonalScore

עבור כל מסמך, עבור על כל מילה המופיעה בו. המידע הרלוונטי על המילה מופיע בקובץ ה-Posting. קריאתו בחזרה ל-RAM מפורט תחת הסעיף של "שימוש בקבצי Posting ושילובם עם האלגוריתם.

להלן המשקלים שאנו נותנים עבור כל פרמטר ששמרנו על המילה:

אם המילה מופיעה בכותרת (isHeader=1) הוסף 1 לדירוג המסמך.

אם המילה מופיעה ב-25 אחוזים הראשונים של המסמך, הוסף 1 לדירוג המסמך.

לאחר מכן, כדי לחשב דירוג סופי עבור המסמכים אנו מבצעים את הנוסחה הבאה:

$$\alpha * BM25 * (1-\alpha) * Personal$$

כאשר, לאחר מספר ניסויים אמפיריים, הגענו למסקנה כי הביצועים הטובים ביותר מגיעים כאשר $a = 0.65$.

כאשר המשתמש מחליט להשתמש בסמנטיקה המשקל משתנה ל- $a = 0.8$, אך כמות המילים שידורגו יהיה גבוה יותר.

אלגוריתם למציאת חמש הישויות הדומיננטיות במסמך

אלגוריתם הדירוג של הישויות החזקות במסמך עובד בצורה הבאה:

כאשר :

מספר הפעמים שמופיעה מילה i במסמך j f_{ij}

Dj – אורך המסמך j

N – מספר המסמכים במאגר

df_i– בכמה מסמכים מופיעה המילה במאגר כולו

$$idf_i = \log_{10} \frac{N}{df_i}$$

$$tf_{ij} = \frac{f_{ij}}{|Dj|}$$

עבור כל ישות במסמך החישוב הינו –

$$\mathrm{R}ank = \ idf_i * tf_{ij}$$

להלן דוגמאות:

במסמך *FBIS3-60299* הישויות החזקות הינן:

Environment Protection Ministr : 0.025
Viktor Kutsenk : 0.023
Environment Protectio : 0.023
Roman Zadunayski : 0.022
Far Eas He : 0.021

במסמך *LA80690-0002* הישויות החזקות הינן:

The NRC : 0.05
Radiological Protection : 0.014
The Nuclear Regulatory Commission : 0.012
International Commission : 0.011
Environmental Protection Agenc : 0.01

אלגוריתם לשיפור סמנטי

להלן הפסאודו קוד:

1. לוקחים מילה מהשאילתה ובודקים האם היא מופיעה במאגר הסמנטי
   1.1 אם כן – לוקחים את שלוש המילים הדומות לה ביותר מבחינה סמנטית ומעבירים אותן להמשך תהליך הדירוג.
   1.2 אחרת חוזרים לשלב 1

לסיכום – בתהליך זה אנו מרחיבים את כמות המילים שאנו מדרגים באמצעותן. מטרת הסמנטיקה, היא למצוא מסמכים בעלי מכנה משותף מבחינת עיסוק, אולם אינם מכילים בדיוק את המילים שאנו מחפשים. כך, אנו יכולים למצוא מסמכים רלוונטיים נוספים מבלי שנאלץ לחשוב על כל המילים האפשרויות שיכולות להופיע במסמך.

שימוש בקבצי Posting ושילובם עם האלגוריתם

המילון אשר נכתב לדיסק מכיל את הנתונים הבאים:

- המילה
- מספר המסמכים בהם המילה מופיעה
- אינדקס בקובץ ה-Posting

קבצי ה-Posting מכילים את המידע הבא:

- האינדקס בקובץ ה-Posting
- שמות המסמכים בהם המילה מופיעה
- כמות המופעים של המילה במסמך
- האם המילה נמצאת בכותרת המסמך
- האם המילה מתחילה באות גדולה
- המיקום היחסי של המילה במסמך

פסאודו קוד לשימוש בקובץ ה-Posting בחישוב הרלוונטיות של מסמך לשאילתה:

1. חפש את המילה במילון
   1.1 אם המילה מופיעה – קח את האינדקס שלה מהמילון

1.1.1 גש לקובץ ה-Posting המתאים

1.1.2 כל עוד לא קראת את השורה אשר מכילה את האינדקס, המשך לקרוא

1.1.3 אם מצאת – פצל את השורה עפ"י המפרידים שהוגדרו מראש

1.1.3.1 המר את מספר המופעים, האם המילה בכותרת, והמיקום היחסי למספרים

1.1.3.2 שלח את המספרים לפונקציית הדירוג להמשך חישוב

1.2 אם המילה לא מופיעה, חזור ל-1

עפ"י הפסאודו קוד, ניתן לראות כי באמצעות שמירת ה-Posting והמילון אנו יכולים לשחזר את המידע על כל מסמך, בדגש על המילים המופיעות בו, מיקומן בתוך המסמך וכמות המופעים שלהן. עם פרטים אלה אנו יכולים לדרג את הרלוונטיות של המסמכים לשאילתה של המשתמש.

פסאודו קוד לשימוש בקובץ ה-Posting והמילון לחישוב חוזק של ישות:

1. חפש את המילה במילון

1.1 אם המילה מופיעה – קח את האינדקס שלה מהמילון

1.1.1 גש לקובץ ה-Posting המתאים

1.1.2 כל עוד לא קראת את השורה אשר מכילה את האינדקס, המשך לקרוא

1.1.3 אם מצאת – פצל את השורה עפ"י המפרידים שהוגדרו מראש

1.1.3.1 עבור על כל הערכים של כמות המופעים של מילה במסמך וסכום אותן

1.1.3.2 העבר את התוצאה לפונקציית הדירוג

1.1.4 גש למילון ותיקח את הערך של כמות המסמכים בהן המילה מופיעה

1.1.5 שלח את הערך לפונקציית הדירוג

1.2 אם המילה לא מופיעה, חזור ל-1

עפ"י הפסאודו קוד, ניתן לראות כי באמצעות שמירת ה-Posting והמילון, אנו יכולים לשחזר את המידע על כמות המופעים של המילה בכל המאגר, וכמות המסמכים שבהן היא מופיעה בכל המאגר. עם נתונים אלה אנו יכולים לחשב את החוזק של ישות בכל המאגר.

שימוש בקוד פתוח

חישוב סמנטיקה נלקח מקוד פתוח.

קישור לאתר - https://github.com/medallia/Word2VecJava

החישוב הסמנטי מבוצע אם המשתמש חפץ בכך, באמצעות סימון במקום המתאים ב-GUI.

ב-Ranker, בפונקציית דירוג המסמכים, יש אפשרות לחפש מילים נרדפות למילה מתוך השאילתה מתוך מאגר שחושב קודם לכן (חלק מהקוד הפתוח בגיט). לאחר מכן, באמצעות מתן משקלים למילים הנרדפות, ניתן למצוא מסמכים רלוונטיים נוספים שעוסקים באותו הנושא, אך אינם מכילים את המילה מהשאילתה שהמשתמש הכניס.

מגישים: איתי כץ 312178312
תומר זיינפלד 312543929

הערכה של המנוע

עם Stemming –

מספר השאילתה	מילות השאילתה	Precision לשאילתה	Recall לשאילתה	Precision @5	Precision @15	Precision @30	Precision @50	משך זמן ריצה בשניות
351	Falkland petroleum exploration	0.24	0.25	0.2	0.2	0.2667	0.24	6
352	British Chunnel impact	0.14	0.028	0.2	0.133	0.2	0.14	3
358	blood-alcohol fatalities	0.34	0.33	0.2	0.4	0.333	0.34	27
359	mutual fund predictors	0.08	0.14	0	0.133	0.133	0.08	21
362	human smuggling	0.1	0.12	0.2	0.133	0.133	0.1	5
367	piracy	0.18	0.04	0	0	0.1	0.18	3
373	encryption equipment export	0.32	0.437	0.2	0.33	0.233	0.32	6
374	Nobel prize winners	0.5	0.12	0.8	0.667	0.5667	0.5	4
377	cigar smoking	0.4	0.55	0	0.2	0.2667	0.4	4
380	obesity medical treatment	0.06	0.42	0	0.133	0.1	0.06	5
384	space station moon	0.16	0.15	0.2	0.133	0.133	0.16	3
385	hybrid fuel cars	0.42	0.24	0	0	0.3	0.42	4
387	radioactive waste	0.22	0.15	0.2	0.2667	0.2333	0.22	3
388	organic soil enhancement	0.16	0.16	0.2	0.2	0.2	0.16	4
390	orphan drugs	0.32	0.13	0.2	0.33	0.43	0.32	4
0.0656 - MAP								

סיכום -

מנוע:

Recall – 0.139

Precision – 0.23

מגישים: איתי כץ 312178312
תומר זיינפלד 312543929

בלי Stemming –

מספר השאילתא	מילות השאילתא	Precision לשאילתא	Recall לשאילתא	Precision @5	Precision @15	Precision @30	Precision @50	משך זמן ריצה
351	Falkland petroleum exploration	0.16	0.16	0	0.2	0.133	0.16	7
352	British Chunnel impact	0.14	0.02	0.2	0.2	0.2	0.14	5
358	blood-alcohol fatalities	0.38	0.37	0.2	0.4667	0.4	0.38	5
359	mutual fund predictors	0.06	0.1	0	0.133	0.1	0.06	18
362	human smuggling	0.12	0.15	0.4	0.133	0.1667	0.12	4
367	piracy	0.18	0.04	0	0	0.1	0.18	4
373	encryption equipment export	0.14	0.43	0.2	0.2	0.23	0.14	5
374	Nobel prize winners	0.46	0.11	0.6	0.6	0.53	0.46	4
377	cigar smoking	0.24	0.33	0	0.06	0.133	0.24	4
380	obesity medical treatment	0.06	0.4	0	0.133	0.1	0.06	4
384	space station moon	0.16	0.15	0.2	0.133	0.1	0.16	4
385	hybrid fuel cars	0.44	0.25	0	0.06	0.33	0.44	5
387	radioactive waste	0.14	0.09	0.4	0.2	0.133	0.14	4
388	organic soil enhancement	0.18	0.18	0	0.133	0.233	0.18	4
390	orphan drugs	0.24	0.09	0.4	0.33	0.3	0.24	3
0.0526 - MAP								

סיכום -

מנוע:

Recall – 0.124

Precision – 0.206

מגישים: איתי כץ 312178312
תומר זיינפלד 312543929

סיכום

בעיות שנתקלנו איתן ודרכים לפתרון

אחת הבעיות המרכזיות שנתקלנו בהן בחלק ב' של המנוע היה אחזור איטי של השאילתות. תחילה, השתמשנו במבנה נתונים שהחזיק כל מילה ואיזה מסמכים מכילים אותה. לאחר מכן, הבנו שדרך יעילה יותר תהיה להפוך את מבנה הנתונים לכזה שיחזיק כל מסמך כמפתח, והערכים יהיו המילים. כך, הצלחנו לשפר בצורה מיטבית את זמני הריצה.

בעיה נוספת שהשפיעה לנו על זמן הריצה, היא יצירה של הרבה אובייקטים זמניים במקום ליצור שדה שמאתחלים פעם אחת, וכן מימוש שגרם הרבה גישות לדיסק. לאחר שזיהינו זאת, יצרנו אובייקטים רבים כשדות, שאותחלו פעם אחת ואם היה צורך בוצע עליהם clear בין האיטרציות.

האתגר הגדול ביותר בפרויקט

האתגר הגדול ביותר בפרויקט הוא ההתנהלות מול כמויות רבות של מידע. בפעם הראשונה בתואר הבנו כמה משמעותית היא בחירה של מבנה נתונים, תכנון אלגוריתמים, צמצום גישות לדיסק ועוד משפיעים על זמן הריצה. כמו כן, התמודדות עם מידע רב יוצר הרבה "רעש", שעם חלקו לעיתים קשה להתמודד, מכיוון שקשה לזהות את מקור הבעיה.

בהמשך, כאשר העבודה הכילה מספר רב של אובייקטים שתיקשרו ביניהם, זה הוסיף אתגר לבדיקת הקוד שכן היה צריך לשים לב למבני נתונים רבים ולשינויים שבוצעו.

המלצות לשיפור האלגוריתם

היינו מנסים למצוא אלגוריתם סמנטי טוב יותר ממה שמצאנו בקוד הפתוח.